



TUGAS AKHIR - TE 145561

**RANCANG BANGUN PENDETEKSI KEBOCORAN DENGAN
MENGUNAKAN *WATER FLOW* SENSOR BERBASIS WI-FI**

Ronaldo Gabe Manik
NRP 103115000100046

Dosen Pembimbing
Ir. Josaphat Pramudijanto, M. Eng.
Dwi Lastomo, S.Si, M.T.

Departemen Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018



FINAL PROJECT - TE 145561

LEAK DETECTOR USING WATER FLOW SENSOR VIA WI-FI

Ronaldo Gabe Manik
NRP 10311500010046

Supervisor
Ir. Josaphat Pramudijanto, M. Eng.
Dwi Lastomo, S.Si, M.T.

ELECTRICAL AUTOMATION ENGINEERING DEPARTMENT
Faculty of Vocational
Institut Teknologi Sepuluh Nopember
Surabaya 2018

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Rancang Bangun Deteksi Kebocoran Menggunakan *Water Flow* Sensor berbasis WIFT**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 21 Juni 2018



Ronaldo Gabe Manik.
NRP 10311500010046

-----Halaman ini sengaja dikosongkan-----

**RANCANG BANGUN PENDETEKSI KEBOCORAN
MENGUNAKAN WATER FLOW SENSOR BERBASIS WIFI**

TUGAS AKHIR


Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik

Pada


Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I


Ir. Josaphat P. M. Eng
NIP. 196210051990031003

Dosen Pembimbing II


Dwi Lastomo S.Si M.T.
NIP. 1987201711059

**SURABAYA
JULI, 2018**

-----Halaman ini sengaja dikosongkan-----

RANCANG BANGUN PENDETEKSI KEBOCORAN MENGUNAKAN *WATER FLOW* SENSOR BERBASIS WIFI

Nama : Ronaldo Gabe Manik
Pembimbing : Ir. Josaphat Pramudijanto, M. Eng.
Dwi Lastomo, S.Si, M.T.

ABSTRAK

Jaringan pipa merupakan transportasi fluida-fluida penting seperti air, minyak dan gas. Memantau jaringan pipa air bawah tanah lebih sulit daripada memantau jalur pipa air yang berada di atas tanah di ruang terbuka. Keadaan ini akan menyebabkan kerugian permanen jika terjadi gangguan pada pipa seperti kebocoran. Kebocoran pada pipa terdiri dari dua jenis yaitu kebocoran non fisik dan fisik. Faktor-faktor yang menyebabkan kebocoran yaitu umur pipa, karat, dan faktor alam.

Cara mendeteksi kebocorannya menggunakan sistem mekanika fluida dan fisika kinematika berdasarkan data laju air yang didapatkan dari *water flow sensor* meter dan disambungkan dengan mikrokontroler. Ketika mendapatkan data debit air pada pipa distribusi nantinya akan diketahui lokasi letak kebocoran dari pipa tersebut, kemudian prototype akan di pasang dengan *water flow sensor* di empat titik pipa yang terletak di ujung dan ditengah-tengah pipa, sehingga dapat mengetahui letak kebocorannya. Di sistem distribusi air ini data akan di olah di Arduino *shield* yang nantinya akan di upload di WiFi. Sehingga dapat di-*monitoring* melalui WiFi tanpa harus ketempat lokasi kebocoran.

Hasil pengujian dari *prototype* ini yaitu ketika terjadi kebocoran maka nilai *flow* akan mengalami penurunan namun *flow* yang letaknya sebelum kebocoran tidak terjadi penurunan, lalu rata-rata eror keempat sensor adalah 2,1735%. Kemudian kemampuan *prototype* ini dalam mendeteksi kebocoran sebesar 0%-18%.

Kata Kunci: WI-FI, *Water Flow Sensor*, *LabView*, *Kebocoran*.

-----Halaman ini sengaja dikosongkan-----

LEAK DETECTOR USING WATER FLOW SENSOR WITH WIFI

Name : Ronaldo Gabe Manik
Supervisor : Ir. Josaphat Pramudijanto, M. Eng.
Dwi Lastomo, S.Si, M.T.

ABSTRACT

Water distribution is generally installed through underground pipes. Network monitor underground water pipes more difficult than monitoring pipelines of the water that was on the ground in open space. This situation will cause a permanent loss in case of disorders of the pipe as it leaks. A leak in the pipe can be caused by several factors, such as age of pipelines, corrosion, incorrect installation, and natural disasters. Therefore, the solution is required to detect a leak in pipe .

How to detect the leaks using a system of fluid mechanics and physics kinematics based on data rate of water obtained from the water flow sensor and connected with microcontroller. When getting data on water distribution pipes discharge will be known the location of leaks from the pipe, then the prototype would be in pairs with water flow sensor in four-point pipe at the end and middle of the pipe, so it can know the leak location. In this water distribution system data will be in processed in Arduino Shield that will be uploaded with Wi-Fi. So it can be monitored via Wi-Fi without must go to the place leak location.

The test results from this prototype is when the leakage occurs then the value of flow will decrease but the flow is located before the leak does not occur decrease, then the average error of the four sensors is 2.1735%. Then the ability of this prototype in detecting leakage of 0%-18%.

Keywords : Wi-Fi, Water Flow sensor, LabView, Leak.

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Segala syukur dan puji hanya bagi Tuhan , oleh karena anugerah-Nya yang melimpah, kemurahan dan kasih setia yang besar. Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Diploma-3 pada Departemen Teknik Elektro Otomasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember Surabaya Dengan terselesaikannya Tugas Akhir ini, Penulis menyampaikan terima kasih yang sebesar - besarnya kepada :

1. Kedua orang tua dan abang yang senantiasa mendoakan dan memberikan semangat dengan tiada henti.
2. Keluarga PSDM ASIK yang senantiasa mendukung dan memberikan semangat dalam mengerjakan tugas akhir saya.
3. Bapak Ir. Josaphat Pramudijanto, M.Eng. dan Bapak Dwi Lastomo, S.Si, M.T. selaku dosen pembimbing 1 dan 2.
4. Pihak Balai Latihan Kerja (BLK) Disnaker Surabaya yang sudah memberi dukungan fasilitas
5. Teman - teman Elektro Industri Angkatan 2015 yang selalu memberikan doa, semangat, dan dukungannya.
6. Teman - teman Hydra Angkatan 2015 yang selalu memberikan doa, semangat, dan dukungannya.
7. Teman-teman Kontrakan Ceria yang menemani dikala suka dan duka.
8. Semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam proses penyelesaian Tugas Akhir ini.

Penulis menyadari dan memohon maaf atas segala kekurangan pada Tugas Akhir ini. Akhir kata, semoga Tugas Akhir ini dapat berman

Surabaya, **21 Juni 2018**

Penulis

-----Halaman ini sengaja dikosongkan-----

DAFTAR ISI

HALAMAN

HALAMAN JUDUL	i
HALAMAN JUDUL	ii
LEMBAR PENGESAHAN	vii
ABSTRAK.....	ix
<i>ABSTRACT</i>	xi
KATA PENGANTAR	xiii
DAFTAR ISI	xv
DAFTAR GAMBAR.....	xvii
DAFTAR TABEL	xix
 BAB I PENDAHULUAN.....	 1
1.1 Latar Belakang	1
1.2 Batasan Masalah	2
1.3 Tujuan	2
1.4 Sistematika Laporan.....	2
1.5 Relevansi.....	3
 BAB II TEORI DASAR PENUNJANG.....	 5
2.1 Penelitian Terkait	5
2.2 Teori Dasar Fluida	7
2.2.1 Dinamika Fluida	7
2.2.2 Persamaan Kontinuitas.....	7
2.2.3 Persamaan Bernoulli	8
2.3 <i>Sensor Water Flow</i> YF-S201	9
2.4 Arduino UNO	10
2.5 <i>W-5100 Ethernet Shield</i>	11
2.6 LabView.....	12
2.7 OPC	12
2.7.1 <i>OPC Server</i>	13
2.8 LCD 16X2.....	13
 BAB III PERANCANGAN ALAT	 15
3.1 Blok Fungsional Sistem	15
3.1.1 Perancangan <i>Prototype</i> Simulasi Kebocoran	16
3.1.2 Konfigurasi Arduino UNO dengan <i>Waterflow</i> Sensor ...	17
3.1.3 Konfigurasi Arduino UNO dengan <i>Ethernet Shield</i>	18

3.2 Perancangan Perangkat Lunak (<i>Software</i>)	18
3.2.1 Pemrograman Pembacaan <i>Water Flow Sensor</i>	18
3.2.2 Perancangan <i>Software LabView</i>	21
BAB IV PENGUJIAN DAN ANALISA ALAT	23
4.1 Kalibrasi <i>Water Flow Sensor</i>	23
4.2 Pengukuran <i>Water Flow Sensor</i> Satu Titik	26
4.3 Pengujian <i>Stop Valve</i>	27
4.4 Pengujian Kebocoran	28
4.5 Pengujian Koneksi <i>Ethernet</i>	32
4.5.1 Pengujian pada <i>Board Arduino UNO</i>	32
4.6 Pengujian Data <i>Telemetry</i>	36
BAB V PENUTUP	41
5.1 Kesimpulan	41
5.2 Saran	41
DAFTAR PUSTAKA	43
LAMPIRAN A	A-1
LAMPIRAN B	B-1
LAMPIRAN C	C-1
RIWAYAT HIDUP	D-1

DAFTAR GAMBAR

Gambar 2.1	Sketsa Pengujian <i>Differential Pressure Tranduser</i>	5
Gambar 2.2	Hasil Analisa Perbedaan Tekanan	6
Gambar 2.3	Sketsa Pemodelan Sistem	6
Gambar 2.4	Pemodelan Sistem Air Tunak	7
Gambar 2.5	Sketsa Pemodelan Bernoulli	8
Gambar 2.6	Bentuk Fisik <i>Water Flow Sensor</i>	9
Gambar 2.7	Bentuk Fisik <i>Ethernet Shield W-5100</i>	11
Gambar 2.8	Diagram Penjelasan <i>OPC Server</i>	12
Gambar 2.9	Penjelasan Hubungan <i>OPC Dengan OPC Client</i>	13
Gambar 2.10	Bentuk Fisik LCD 16x2	13
Gambar 2.11	Sistem Distribusi PDAM	14
Gambar 3.1	Sketsa Fungsional Sistem	15
Gambar 3.2	Sketsa Rancangan <i>Prototype</i> Tampak Depan	16
Gambar 3.3	Sketsa <i>Prototype</i> Tampak Samping	17
Gambar 3.4	Konfigurasi Arduino UNO dengan <i>Water Flow</i>	17
Gambar 3.5	Konfigurasi <i>Ethernet Shield</i> dengan Arduino UNO	18
Gambar 3.6	<i>Flowchart</i> Program Arduino.	19
Gambar 3.7	<i>Flowchart</i> NIOPC	20
Gambar 3.8	<i>Block Diagram</i> LabView	21
Gambar 3.9	<i>Front Panel</i> LabView	22
Gambar 4.1	Menghubungkan <i>Water Flow</i> Dengan Z-3003	23
Gambar 4.2	Pengujian Menggunakan <i>Prototype</i>	28
Gambar 4.3	Program Pada Arduino UNO ke-1	33
Gambar 4.4	<i>Command Prompt</i> Pada Komputer Dengan IP Address Arduino UNO ke-1	33
Gambar 4.5	Program Pada Arduino UNO ke-2	34
Gambar 4.6	<i>Command Prompt</i> pada Arduino 2	34
Gambar 4.7	Program Arduino ke- 3	34
Gambar 4.8	<i>Command Prompt</i> Arduino ke-3	35
Gambar 4.9	Program Arduino ke-4	35
Gambar 4.10	<i>Command Prompt</i> Arduino ke-4	35
Gambar 4.11	Tampilan Awal LabView Terhubung Dengan Wi-Fi	36
Gambar 4.12	Tampilan Kondisi Normal Tanpa Kebocoran	37
Gambar 4.13	Tampilan Kondisi Bocor $\pm 20^\circ$ di Titik 1.	37
Gambar 4.14	Tampilan Kondisi Bocor $\pm 30^\circ$ di Titik 1.	38

Gambar 4.15 Tampilan Kondisi Bocor $\pm 20^\circ$ di Titik 2 38

Gambar 4.16 Tampilan Kondisi Bocor $\pm 30^\circ$ di Titik 2 39

Gambar 4.17 Tampilan Kondisi Bocor $\pm 20^\circ$ di Titik 3 39

Gambar 4.18 Tampilan Kondisi Bocor $\pm 30^\circ$ di Titik 3 40

DAFTAR TABEL

	HALAMAN
Tabel 2.1 Spesifikasi <i>Water Flow</i> Sensor	10
Tabel 2.2 Spesifikasi Arduino UNO	11
Tabel 4.1 Kalibrasi <i>Water Flow</i> Sensor 1	24
Tabel 4.2 Kalibrasi <i>Water Flow</i> Sensor 2	24
Tabel 4.3 Kalibrasi <i>Water Flow</i> Sensor 3	25
Tabel 4.4 Kalibrasi <i>Water Flow</i> Sensor 4	26
Tabel 4.5 Pengujian <i>Water Flow</i> Sensor 1 Titik yang Sama	26
Tabel 4.6 Keadaan <i>Stop Valve</i> 22,5°	27
Tabel 4.7 Keadaan <i>Stop Valve</i> 45°	27
Tabel 4.8 Keadaan <i>Stop Valve</i> 53°	27
Tabel 4.9 Saat Keadaan Normal	29
Tabel 4.10 Saat Keadaan Kran Kebocoran 1 Dengan Sudut $\pm 20^\circ$.	29
Tabel 4.11 Saat Keadaan Kran Kebocoran 1 Dengan Sudut $\pm 30^\circ$.	29
Tabel 4.12 Saat Keadaan Kran Kebocoran 1 Dengan Sudut $\pm 45^\circ$.	30
Tabel 4.13 Saat Keadaan Kran Kebocoran Dengan Sudut $\pm 20^\circ$	30
Tabel 4.14 Saat Keadaan Kran Kebocoran 2 Dengan Sudut $\pm 30^\circ$.	30
Tabel 4.15 Saat Keadaan Kran Kebocoran 2 Dengan Sudut $\pm 45^\circ$.	31
Tabel 4.16 Saat Keadaan Kran Kebocoran 3 Dengan Sudut $\pm 20^\circ$.	31
Tabel 4.17 Saat keadaan Kran Kebocoran 3 Dengan Sudut $\pm 30^\circ$..	31
Tabel 4.18 Saat Keadaan Kran Kebocoran 3 Dengan Sudut $\pm 45^\circ$.	32

-----Halaman ini sengaja dikosongkan-----

BAB I

PENDAHULUAN

1.1 Latar Belakang

Salah satu sumber daya alam yang sangat dibutuhkan oleh manusia dan merupakan kebutuhan pokok adalah air. Penggunaan air bersih pada kehidupan sehari-hari sangat berdampak kepada masyarakat sedangkan air merupakan unsur penting bagi setiap organisme, kebutuhan untuk menyediakan distribusi air yang baik sehingga sistem adalah suatu keharusan. Terkadang, kondisi di lokasi tertentu tidak memungkinkan kita menciptakan air yang baik di sistem distribusi di lapangan dan pengembangan konstruksi menyebabkan air saat ini sistem distribusi ke kantor residensial, perkantoran, dan industri melalui pipa di bawah tanah. Kerugian akibat kebocoran air menjadi perhatian utama jaringan air. Bahkan kebocoran kecil pun bisa berakibat fatal dalam hilangnya ribuan liter air jika dibiarkan tidak terdeteksi. Mengingat bahwa setiap liter air yang terbuang ini telah diperbaiki dan energi telah yang dikeluarkan untuk memperbaiki pipa-pipa yang ada di sekitar jaringan, kerugian tersebut juga akan merugikan operator air dan ada bukti untuk membenarkan hal itu. Tanggal 03 Januari 2017 bahwa 3.000 ribu masyarakat Surabaya kesulitan dan mendapatkan air karena terdapat kebocoran dan melakukan perbaikan pipa yang bocor di daerah MERR Rungkut.[1]

Pipa PDAM yang terdapat di Surabaya mencapai 37.000 Km². Dalam mendeteksi kebocoran pipa, PDAM menggunakan sistem DMA (*Distric Metered Area*), yaitu dengan sistem yang dititik-titiknya terdapat *water flow* sensor elektromagnetik, alat tersebut akan mengirim data *flow* di setiap tersebut dengan menggunakan *data logger*. Jika terjadi ketidak normalan, dapat dilihat dari data-data *flow* yang didapatkan dan dianalisa, penyebab ketidak normalan *flow* tersebut bisa disebabkan berbagai hal seperti meteran rusak ataupun kebocoran untuk menangani hal tersebut dipegang oleh departemen penanggulangan kehilangan air, contoh kasus di daerah Klampis yang mengalami kebocoran seperti yang terlampir pada lampiran A-2. Dapat diketahui bahwa alat ukur *flow* yang digambarkan kotak biru. Ketika terjadi ketidak normalan, petugas dapat memeriksa lokasi tersebut. Kedepannya alat ini dapat diimplementasikan pada pipa utama dan bisa juga diletakkan pipa distribusi air ke konsumen.

Sensor yang akan digunakan adalah *water flow* sensor meter yang akan bekerja ketika diberikan tegangan 5-15 DC Volt, dan mampu

mendeteksi air dengan memiliki tekanan maksimu 0,8MPa. Ketika data sudah didapatkan melalui *water flow sensor* akan dikirim oleh mikrokontroler ke *LabView* via WiFi. Nantinya, kita dapat melihat posisi letak kebocoran melalui perbedaan debit di antara masing-masing titik *water flow sensor* dan perbedaan debit air di setiap *water flow sensor* akan ditampilkan dalam grafik dengan melalui *LabView* lalu melalui grafik yang ditampilkan melalui *LabView* maka akan di tampilkan oleh web sehingga pengguna dapat dimudahkan mendeteksi letak kebocoran dimanapun dan secara *real time*.

1.2 Perumusan Masalah

Permasalahan yang dihadapi pada Tugas Akhir ini adalah sulitnya mendeteksi kebocoran pada pipa. Kebocoran pipa bila dibiarkan secara terus menerus dapat menyebabkan kerugian karena banyak air yang akan terbuang sia-sia.

1.3 Batasan Masalah

Pada Tugas Akhir ini, memiliki batasan-batasan masalah yang diambil, diantaranya :

1. Alat ini adalah *prototype/rancangan* alat deteksi kebocoran dengan menggunakan sensor *water flow*.
2. Sensor hanya mampu membaca debit yang berukuran kecil (batas sensor).
3. Air pada saluran didapatkan dari pompa air.
4. Hanya bisa mendeteksi di satu titik kebocoran .

1.4 Tujuan

Tujuan dari penelitian ini adalah :

1. Dapat mengetahui lokasi titik kebocoran pada pipa .
2. Merancang *monitoring* pipa dengan komunikasi berbasis Wi-Fi.
3. Merancang dan membuat *prototype* sistem distribusi air.

1.5 Sistematika Laporan

Pembahasan Tugas Akhir ini dibagi menjadi lima bab dengan sistematika sebagai berikut:

Bab I Pendahuluan

Bab ini meliputi latar belakang, permasalahan, tujuan penelitian, metodologi penelitian, sistematika laporan, dan relevansi.

Bab II Teori Penunjang

Bab ini menjelaskan tentang tinjauan pustaka, konsep dari *Arduino* UNO, teori dasar fluida, sensor *water flow*, *W-5100*, *LabView*, *NIOPC*,

Bab III Perancangan Sistem

Bab ini membahas perencanaan dan pembuatan perangkat keras (*Hardware*) yang meliputi desain alat serta pengimplementasian sensor yang digunakan, pengaturan modul *ethernet shield*, dan pembuatan perangkat lunak (*Software*) yang meliputi program pada *Arduino IDE* untuk menjalankan alat tersebut, serta pembuatan HMI (*Human Machine Interface*) dengan menggunakan *LabView*

Bab IV Simulasi, Implementasi dan Analisis Sistem

Bab ini memuat tentang pemaparan dan analisis hasil pengujian alat pada keadaan sebenarnya. Seperti pengujian sensor *waterflow*. Selain itu, dilakukan pengujian *Ethernet Shield*, dan pengujian aplikasi *LabView*

Bab V Penutup

Bab ini berisi kesimpulan dan saran dari hasil pembahasan yang telah diperoleh.

1.6 Relevansi

Tugas Akhir ini bertujuan untuk membuat *prototype* sistem yang dapat mendeteksi letak kebocoran dengan menggunakan *water flow sensor* secara akurat dan dapat di-*monitoring* secara *wireless*.

-----Halaman ini sengaja dikosongkan-----

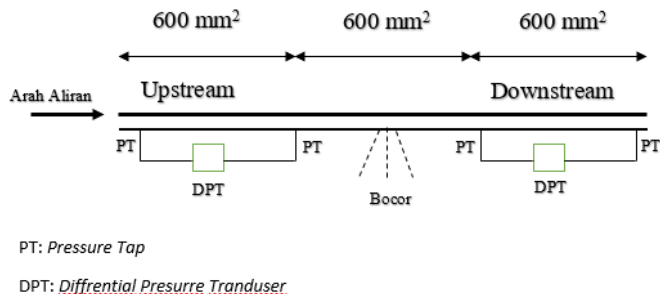
BAB II

TEORI DASAR PENUNJANG

Bab ini membahas mengenai teori dasar dari peralatan yang digunakan dalam Tugas Akhir yang berjudul Rancang Bangun Pendeteksi Kebocoran Menggunakan *Water Flow Sensor* berbasis Wifi. Materi ini digunakan sebagai dasar materi untuk pembuatan alat yang dibuat masing-masing mahasiswa untuk pembuatan keseluruhan alat ini.

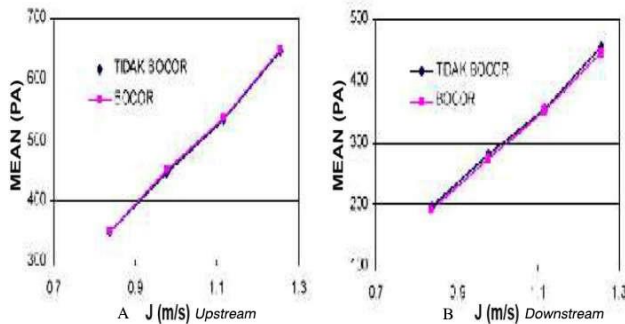
2.1 Penelitian Terkait [2] [3]

Deteksi Kebocoran Pipa Menggunakan Prinsip Beda Tekanan Penelitian terkait deteksi kebocoran pipa dilakukan menggunakan teknologi *Differential Pressure Transducer* (DPT) yang ditempatkan sebelum dan sesudah titik kebocoran untuk merekam beda tekanan. DPT dihubungkan dengan peralatan pengkondisi sinyal dan ADC yang menghasilkan data beda tekanan **Gambar 2.1** menunjukkan sketsa pengujian yang dilakukan.



Gambar 2.1 Sketsa Pengujian *Differential Pressure Transducer*

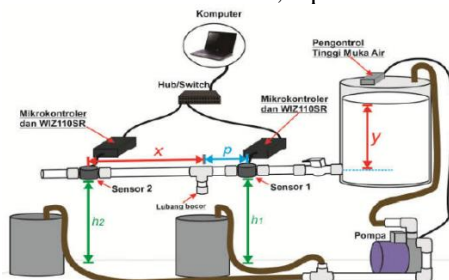
Jarak pengukuran kebocoran adalah 600 mm dan kebocoran dibuat dilakukan dengan menggunakan *solenoid valve*. Dalam penelitian tersebut, pengolahan data menggunakan program Matlab untuk memperoleh grafik beda tekanan dalam rangkaian waktu, *mean*, *probability density function*



Gambar 2.2 Hasil Analisa Perbedaan Tekanan.

Berdasarkan **Gambar 2.2**, pada *upstream*, kondisi bocor tidak menghasilkan perubahan beda tekanan. Pada *downstream*, kondisi bocor menghasilkan beda tekanan lebih tinggi apabila dibandingkan dengan kondisi tidak bocor. Hal ini diakibatkan perubahan tekanan pada titik kebocoran (menjadi tekanan atmosfer). Berdasarkan hasil tersebut diperoleh bahwa pada kondisi pipa bocor menghasilkan beda tekanan lebih tinggi dibandingkan kondisi pipa tidak bocor.

Menurut penelitian Dwi Hariyanto (2016) dengan metode *water flow* sensor model FS300A, beliau menganalisa perbedaan debit air setiap titik. Setelah mengetahui adanya letak kebocoran, sistem ini akan mengolah data yang didapatkan oleh sensor lalu hasil tersebut di transmisikan ke komputer menggunakan jaringan berbasis TCP/IP melalui modul WIZ110SR untuk dianalisa, diperlihatkan **Gambar 2.3**



Gambar 2.3 Sketsa Pemodelan Sistem

2.2 Teori Dasar Fluida

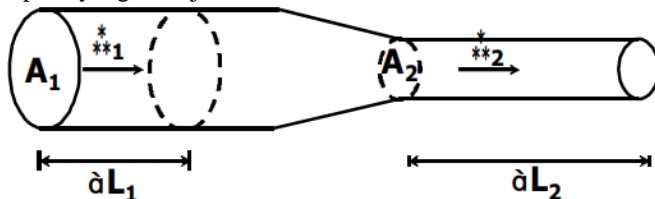
David Halliday (1978) mengatakan bahwa fluida (*fluid*) adalah salah satu zat yang dapat mengalir. Jadi istilah fluida termasuk cairan dan gas. Klasifikasi seperti itu tidaklah terlalu jelas. Beberapa fluida, seperti gelas, mengalir begitu lambat sehingga berperilaku seperti benda padat untuk interval-interval waktu yang biasanya digunakan untuk bekerja dengan benda-benda tersebut. Plasma, yang merupakan gas yang sangat terionisasi tidak cocok untuk digolongkan ke dalam salah satu dari kategori ini; plasma tersebut seringkali dinamakan ‘keadaan keempat materi’ (*fourth state of matter*) untuk membedekannya dari keadaan padat, keadaan cair, dan keadaan gas.

2.2.1 Dinamika Fluida [4][5]

Salah satu cara untuk menjelaskan gerak suatu fluida adalah dengan membagi fluida tersebut menjadi elemen-elemen volume yang sangat kecil, yang dapat kita namakan partikel-partikel fluida, dan mengikuti gerak masing-masing partikel ini. Cara ini adalah sebuah tugas yang berat. Kita akan memberikan koordinat x , y , z kepada setiap partikel fluida seperti itu dan akan menentukan koordinat-koordinat ini sebagai fungsi-fungsi dari waktu t . Ada suatu cara pembahasan, yang dikembangkan oleh Leonhard Euler (1707-1783), yang lebih mudah digunakan untuk kebanyakan tujuan. Didalam pembahasan tersebut, kita menjelaskan gerak fluida dengan menspesifikasikan massa jenis $\rho(x, y, z, t)$ dan kecepatan $v(x, y, z, t)$ di titik (x, y, z) pada waktu t . Jadi kita memusatkan perhatian kita pada apa yang terjadi di sebuah titik khas didalam ruang pada waktu yang khas.

2.2.2 Persamaan Kontinuitas

Suatu fluida yang sedang mengalir melalui suatu volume yang tetap (misalnya sebuah tangki) yang mempunyai satu sisi masuk, dan satu sisi keluar seperti yang ditunjukkan Gambar 2.4

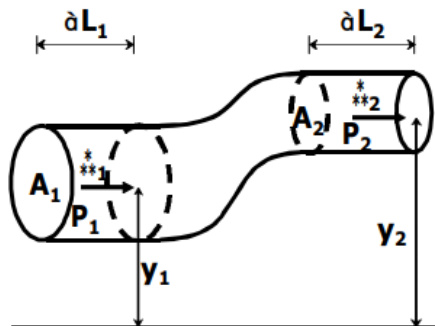


Gambar 2.4 Pemodelan Sistem Air Tunak

Jika alirannya tunak sehingga tidak terjadi akumulasi tambahan fluida dalam volume tersebut, laju aliran fluida yang masuk ke dalam volume harus sama dengan laju aliran yang keluar dari volume (karena kalau tidak massa-nya tidak kekal). Laju aliran massa dari sebuah sisi keluar, m (slug/s atau kg/s), diberikan oleh $m = \rho Q$, dimana Q (ft³/s atau m³/s) adalah laju aliran volume. Jika luas sisi keluar A dan fluida mengalir melintasi luas ini (tegak lurus/normal terhadap luas) dengan kecepatan rata-rata v , maka volume dari fluida yang melintasi sisi keluar ini dalam selang waktu adalah yang sama dengan sebuah volume dengan Panjang $v\delta t$ dan luas penampangnya A . Jadi laju aliran volume (volume per satuan waktu) adalah $Q = vA$. Sehingga, $m = \rho vA$. Untuk massa yang kekal, laju aliran masuk harus sama dengan laju aliran keluar. Jika sisi masuk ditandai dengan 1, dan sisi keluar, maka $m_1 = m_2$.

2.2.3 Persamaan Bernoulli

Aliran dari suatu segmen fluida ideal yang melewati pipa tidak beraturan dalam selang waktu Δt ditunjukkan **Gambar 2.5**



Gambar 2.5 Sketsa Pemodelan Bernoulli

Pada awal selang waktu tersebut, segmen dari fluida terdiri atas bagian yang diarsir (bagian 1) di sebelah kiri dan bagian yang tidak diarsir. Selama selang waktu tersebut, ujung sebelah kirinya bergerak ke kanan sejauh jarak Δx_1 , yang merupakan panjang dari bagian yang diarsir di sebelah kiri. Sedangkan ujung sebelah kanannya bergerak ke kanan sejauh jarak Δx_2 , yang merupakan Panjang dari bagian abu-abu yang diarsir (bagian 2) di bagian kanan atas **Gambar 2.5**. Oleh karena itu, pada akhir dari selang waktu tersebut, segmen fluida terdiri dari bagian yang

tidak diarsir dan bagian abu-abu yang diarsir di sebelah kanan atas. Usaha total yang dilakukan pada sistem oleh fluida di luar segmen sama dengan perubahan energi mekanik sistem $W = \Delta K + \Delta U$. Dengan melakukan substitusi untuk setiap suku dalam persamaan ini, diperoleh:

$$(P_1 - P_2) V = \frac{1}{2} m v^2 \quad P_1 - P_2 V = \frac{1}{2} m v^2 \dots\dots\dots (2.1)$$

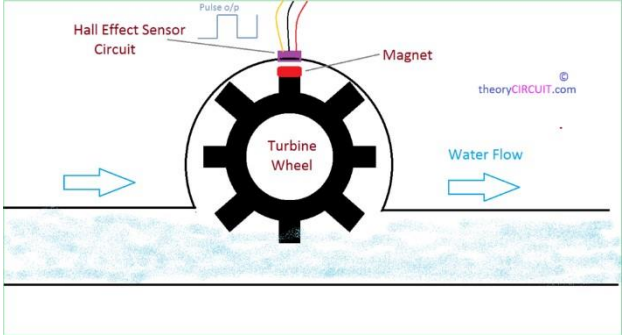
Diatas merupakan persamaan Bernouli sebagaimana dapat diterapkan pada fluida ideal. Persamaan ini juga dapat ditulis:

$$P + \frac{1}{2} \rho v^2 + \rho g y = \text{konstan} \dots\dots\dots (2.2)$$

Persamaan ini menunjukkan bahwa tekanan fluida berkurang ketika kelajuan fluida bertambah.

2.3 Sensor Water Flow YF-S201 [6]

Water flow sensor terdiri dari badan katup plastik, rotor air, dan sensor efek hall. Keunggulan *waterflow* sensor berbasis sensor *effect hall* yaitu sistem deteksinya non-kontak sehingga tahan lama dan keluarannya berupa sinyal *digital* sehingga mudah diproses dan kebal terhadap *noise*. Berikut bentuk fisik *water flow* sensor **Gambar 2.6**.



Gambar 2.6 Bentuk Fisik *Water Flow* Sensor

Ketika air mengalir melalui rotor, rotor berputar. Perubahan kecepatan air mengalir dengan tingkat debit berbeda-beda. Keluaran sensor efek hall berupa sinyal pulsa. Kelebihan sensor ini adalah hanya membutuhkan satu sinyal selain jalur 5V DC dan *Ground*. Spesifikasi *water flow sensor* model YF-S201 adalah sebagai berikut **Tabel 2.1**.

Tabel 2.1 Spesifikasi *Water Flow Sensor*

<i>Min Working Voltage</i>	DC 4,5 V
<i>Max Working Current</i>	15mA(DC 5v)
<i>Working Voltage</i>	5V ~ 24V
<i>Flow Rate Range</i>	1 ~ 60L/min
<i>Load Capacity</i>	≤10mA(DC 5V)
<i>Operating Temperature</i>	≤80°C
<i>Liquid Temperature</i>	≤120°C
<i>Operating Humidity</i>	35% ~ 90% RH
<i>Water Pressure</i>	≤2.0 MPa
<i>Storage Temperature</i>	-25°C ~ +80°C
<i>Storage Humidity</i>	25% ~ 95% RH

2.4 Arduino UNO [7]

Arduino merupakan perangkat elektronik atau papan rangkaian elektronik *open source* yang di dalamnya terdapat komponen utama, yaitu sebuah *chip* mikrokontroler dengan jenis AVR dari perusahaan Atmel. Mikrokontroler itu sendiri adalah *chip* atau IC (*Integrated circuit*) yang bisa diprogram menggunakan komputer. Tujuan menanamkan program pada mikrokontroler adalah agar rangkaian elektronik dapat membaca input, memproses *input* tersebut dan kemudian menghasilkan *output* sesuai yang diinginkan. Jadi mikrokontroler bertugas sebagai otak yang men-gendalikan *input*, proses, dan output sebuah rangkaian elektronik.

Mikrokontroler sendiri adalah sebuah komputer kecil disuatu sirkuit terpadu yang berisi tentang inti prosesor, memori dan *input/output* yang telah diprogram. Program disimpan dalam bentuk Ferroelectric RAM, Nor *Flash*, OTP ROM yang disertakan dalam *chip*. Mikrokontroler digunakan untuk aplikasi *embedded*, tidak seperti mikroprosesor yang digunakan dalam komputer pribadi. Fungsi dari mikrokontroler adalah untuk mengontrol produk atau perangkat secara otomatis seperti sistem kontrol mesin mobil, mesin kantor, alat-alat listrik, dan sistem *embedded* lainnya.

Salah satu jenis Arduino yang menggunakan mikrokontroler tipe Atmel AVR (8-bit) adalah Arduino UNO. Arduino UNO adalah papan mikrokontroler berbasis ATmega 328. Spesifikasinya adalah seperti **Tabel 2.2.**

Tabel 2.2 Spesifikasi Arduino UNO

Mikrokontroler	ATmega328
Operasi tegangan	5 Volt
Input tegangan batas	6-20 Volt
Pin I/O <i>digital</i>	14
Pin <i>Analog</i>	6
Arus DC setiap pin I/O	50mA

2.5 W-5100 Ethernet shield [7]

Arduino *ethernet shield* memungkinkan papan Arduino untuk terhubung ke internet. Ini didasarkan pada *chip ethernet* Wiznet W5100 (*datasheet*). Wiznet W5100 menyediakan jaringan (IP) *stack* yang dapat digunakan baik TCP maupun UDP. Mendukung hingga empat koneksi soket simultan. Gunakan *library ethernet* untuk menulis sketsa yang terhubung ke internet menggunakan *shield*. *ethernet shield* terhubung ke papan Arduino menggunakan *header* kawat-*wrap* panjang yang membentang melalui *shield*. Ini menjaga tata letak pin tetap utuh dan memungkinkan perisai lain ditumpuk di atasnya. Revisi terbaru dari papan memperlihatkan pin *out* 1.0 pada rev 3 *board* Arduino UNO. *ethernet shield* V1 memiliki koneksi RJ-45 standar, dengan transformator garis terintegrasi dan *power over ethernet* diaktifkan. Ada slot kartu *micro-SD onboard*, yang dapat digunakan untuk menyimpan *file* untuk melayani melalui jaringan. Ini kompatibel dengan semua papan Arduino / Genuino. Pembaca kartu *micro SD on board* dapat diakses melalui Perpustakaan SD. Saat bekerja dengan pustaka ini, SS ada di Pin 4. Revisi *ethernet shield* berisi slot kartu SD ukuran penuh; ini tidak didukung. Bentuk fisik *Ethernet shield* seperti **Gambar 2.7**



Gambar 2.7 Bentuk Fisik *Ethernet shield* W-5100

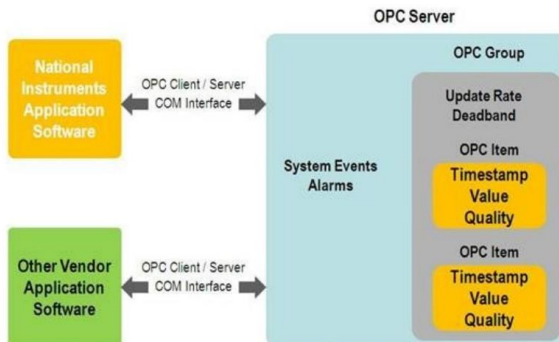
2.6 LabView [8]

LabView adalah sebuah *software* pemrograman yang diproduksi oleh National Instruments yang bahasa pemrogramannya menggunakan *block diagram*, dan aplikasi ini memiliki kemampuan untuk berkomunikasi dengan PLC, dan perangkat lainnya.. Pada LabView, *user* pertama-tama membuat *user interface* atau *front panel* dengan menggunakan kontrol dan indikator, yang dimaksud dengan kontrol adalah *knobs*, *push buttons*, *dials* dan peralatan input lainnya sedangkan yang dimaksud dengan indikator adalah *graphs*, LED dan peralatan *display* lainnya.

Setelah menyusun *user interface*, lalu *user* menyusun *blok diagram* yang berisi kode-kode VIs untuk mengontrol *front panel*. Software LabView terdiri dari tiga komponen utama, yaitu : *front panel*, *Blok diagram* dari Vi, *Control* dan *Functions Pallette*.

2.7 OPC [8]

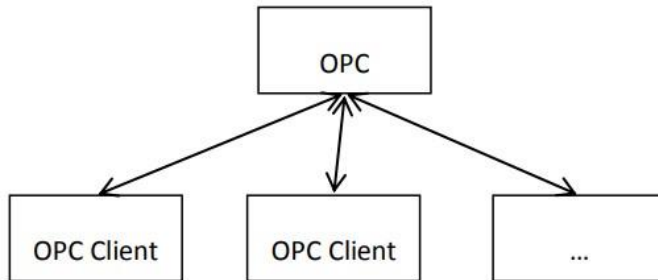
OPC (OLE for *Procces Control*) dan sedangkan kepanjangan OLE adalah bentuk dari tampilan antar muka (*Human Machine Interface*) yang umumnya di gunakan diantara berbagai sumber data yang ada seperti *Programmable Logic Controller* (PLC) , *Remote Terminal Unit* (RTU) dan sensor- sensor yang ada di pabrik melalui HMI / SCADA .Dengan menggunakan OPC, *software* yang anda gunakan sebagai HMI dapat melakukan komunikasi data dengan perangkat keras anda (*hardware*) tanpa harus menduplikat *device driver*, dapat diilustrasikan seperti **Gambar 2.8**.



Gambar 2.8 Diagram Penjelasan OPC Server

2.7.1 OPC Server [8]

Konsep dasar dari OPC *server* adalah kita memiliki OPC *server* dan beberapa OPC *client* untuk berkomunikasi dengan *server* tersebut dan untuk menulis data atau membaca data tersebut. Dapat dicontoh seperti **Gambar 2.9**.

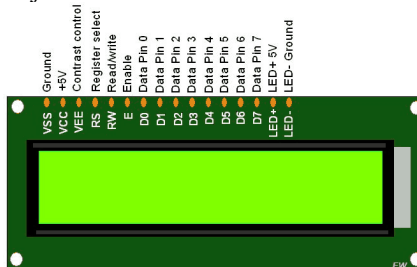


Gambar 2.9 Penjelasan Hubungan OPC Dengan OPC Client

Disaat keadaan tingkat tinggi OPC *server* terbagi beberapa komponen yaitu *server*, grup, dan barang. OPC *server* tersebut mempunyai informasi tentang *server* dan menjadi wadah untuk grup objek OPC.

2.8 LCD 16X2 [9]

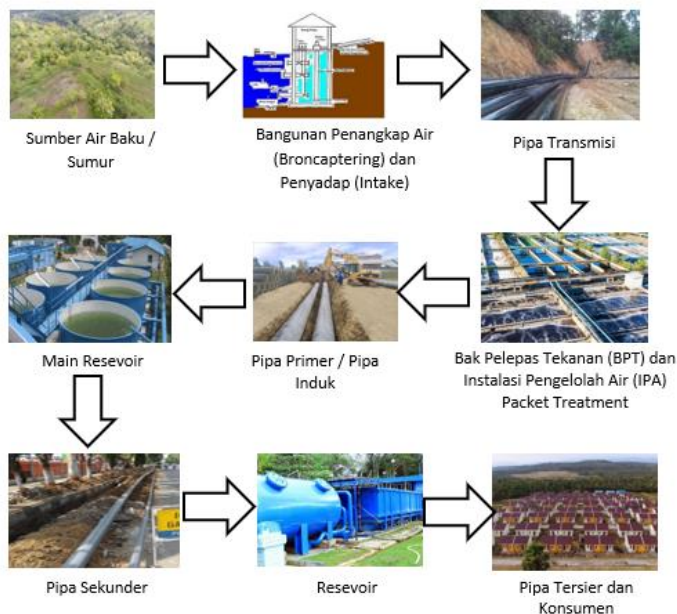
Suatu media tampilan atau *display* yang menggunakan kristal cair sebagai penampil utama. LCD sudah digunakan diberbagai bidang misalnya alat-alat elektronik. LCD berfungsi untuk menampilkan karakter, angka, huruf, ataupun simbol lainnya. Dengan adanya alat ini, sangat membantu untuk memantau keadaan sensor ataupun keadaan jalanya program. Penampilan LCD 16x2 ini bisa dihubungkan dengan mikrokontroler apa saja.



Gambar 2.10 Bentuk Fisik LCD 16x2

2.9 Sistem Distribusi Air [10]

PDAM atau Perusahaan Daerah Air Minum merupakan salah satu usaha milik daerah, yang bergerak dalam distribusi air bagi masyarakat Indonesia .



Gambar 2.11 Sistem Distribusi PDAM

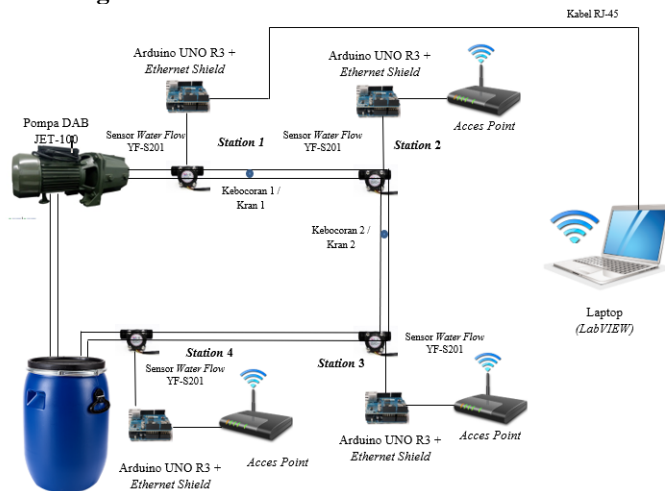
Pada **Gambar 2.11** dapat dilihat sistem distribusi PDAM yang ada di Indonesia. Pertama-tama sumber air baku biasanya didapatkan dari sungai ataupun sumur yang telah digali lalu disalurkan ke bangunan penangkap air dan disalurkan menggunakan pipa transmisi ke bak pelepas tekanan dan instalasi pengelolah air. Kemudian air tersebut didistribusikan dengan pipa induk ke *main reservoir*, setelah itu air dari *main reservoir* disalurkan ke *reservoir* menggunakan pipa sekunder yang nantinya air tersebut didistribusikan ke konsumen menggunakan pipa tersier. Pada alat ini sangat cocok jika dipasang di pipa induk karena lokasinya didalam tanah sehingga dapat memudahkan petugas untuk melakukan *maintenance*. Contoh sistem distribusi air dapat dilihat pada lampiran A-3, dapat dilihat sumber airnya dari mana saja .

BAB III

PERANCANGAN DAN PEMBUATAN ALAT

Pada bab ini akan dibahas mengenai perancangan dan pembuatan alat “Rancang Bangun Pendeteksi Kebocoran Menggunakan *Water Flow* Sensor berbasis Wi-Fi”. Penjelasan diawali dengan penjelasan diagram fungsional sistem secara keseluruhan, kemudian perancangan perangkat keras dan diakhiri perencanaan perangkat lunak. Perancangan perangkat keras (*hardware*) meliputi perancangan *prototype* simulasi kebocoran pipa beserta perangkat elektriknya. Dan perancangan perangkat lunak (*software*) meliputi pemrograman arduino IDE pada arduino UNO untuk pengukuran debit air pada sensor *water flowmeter* serta *monitoring* debit air dengan *interface* LabView.

3.1 Blok Fungsional Sistem



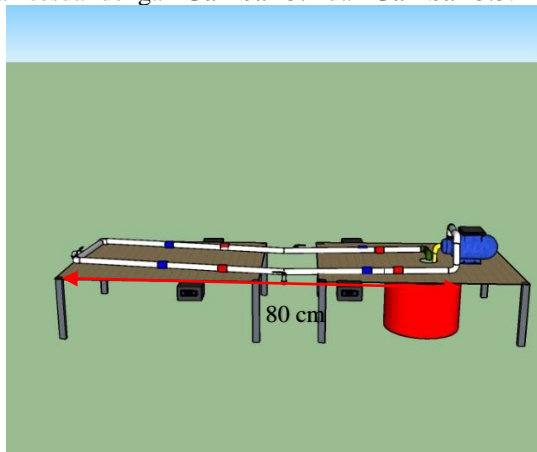
Gambar 3.1 Sketsa Fungsional Sistem

Berdasarkan **Gambar 3.1**, sumber air berasal dari gentong biru yang kemudian dipompa dengan menggunakan pompa air JET-100 ,pada *station 1* terdapat sensor *water flow* yang telah dihubungkan dengan Arduino UNO dan *Ethernet shield* yang nantinya akan disambung dengan

kabel RJ-45 ke Komputer , lalu untuk *station 2* terdapat Arduino UNO berserta *Ethernet shield* yang nantinya dihubungkan dengan *access point* dan sensor *water flow*. Lalu, *station 3* terdapat Arduino UNO beserta *ethernet shield* akan dihubungkan dengan sensor *water flow* dan *access point*. Terakhir , *station 4* terdapat Arduino UNO beserta *Ethernet shield* Akan dihubungkan dengan sensor *water flow* dan *access point*, kemudian ketika air sudah melewati keempat *station* maka air akan dibuang ke gentong biru. Kemudian data diolah ke dalam bentuk tampilan LabView

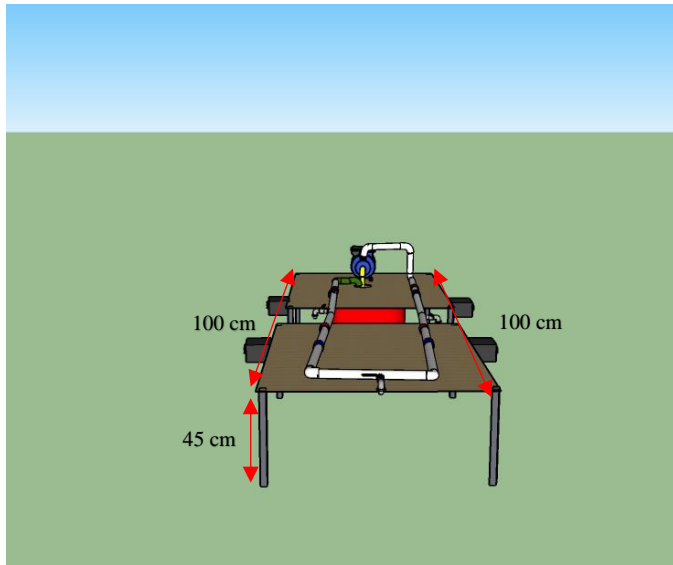
3.1.1 Perancangan *Prototype* Simulasi Kebocoran

Perancangan *prototype* simulasi kebocoran pipa ini dimaksudkan untuk jarak yang jauh dan dengan kondisi lingkungan yang tidak biasa. Terdapat tiang penyangga digunakan untuk menyangga papan triplek tersebut, lalu terdapat meja yang digunakan untuk me-nompang pompa air Jet-100 dan terdapat papan triplek dengan tebal 4 mm² yang akan digunakan untuk wadah pipa tersebut. Kami menggunakan pipa berukuran ½ in untuk simulasi *prototype* ini. Lalu kami mensimulasikan kebocorannya menggunakan kran air, yang nantinya akan menunjukkan nilai kebocoran sesuai dengan **Gambar 3.2** dan **Gambar 3.3**.



Gambar 3.2 Sketsa Rancangan *Prototype* Tampak Depan

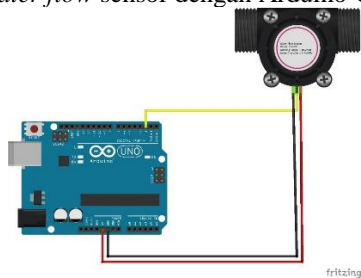
Dapat dilihat bahwa terdapat kotak hitam yang berisi mikrokontroler berserta *ethernet shield*, selain itu terdapat LCD 16x2 yang berfungsi menampilkan pembacaan sensor.



Gambar 3.3 Sketsa *Prototype* Tampak Samping

3.1.2 Konfigurasi Arduino UNO dengan *Waterflow* Sensor

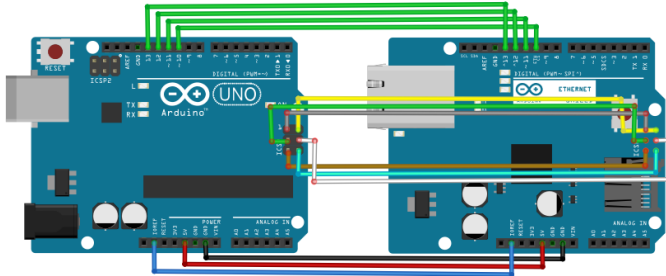
Arduino UNO dihubungkan dengan *Water flow* sensor dengan cara memasukan sinyal dari pin 2, dan di pin ini terdapat fungsi *attach interrupt 0* yang berfungsi untuk membuat semuanya menjadi otomatis. Berikut konfigurasi *water flow* sensor dengan Arduino **Gambar 3.4**.



Gambar 3.4 Konfigurasi Arduino UNO dengan *Water Flow*

3.1.3 Konfigurasi Arduino UNO dengan *Ethernet Shield*

Pada perangkat komunikasi digunakan *ethernet shield* yang telah *compatible* dengan Arduino UNO. Sehingga pemasangan *ethernet shield* pada Arduino UNO hanya dengan digabungkan pada bagian atas Arduino UNO saja. Gabungan antara Arduino UNO dan *ethernet shield* sering dinamakan Arduino *Web Server*. Gabungan *ethernet shield* yang dipasang di atas Arduino UNO dapat dilihat pada **Gambar 3.5**



Gambar 3.5 Konfigurasi *Ethernet Shield* dengan Arduino UNO

3.2 Perancangan Perangkat Lunak (*Software*)

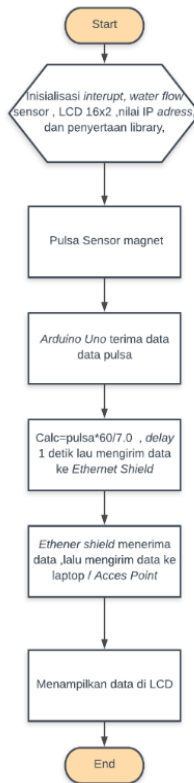
Pada perancangan perangkat *software* pada tugas akhir ini yang dibahas terdiri dari pemrograman pengukuran debit air oleh sensor pada software Arduino IDE, dan perancangan *software* LabView.

3.2.1 Pemrograman Pembacaan *Water Flow Sensor*

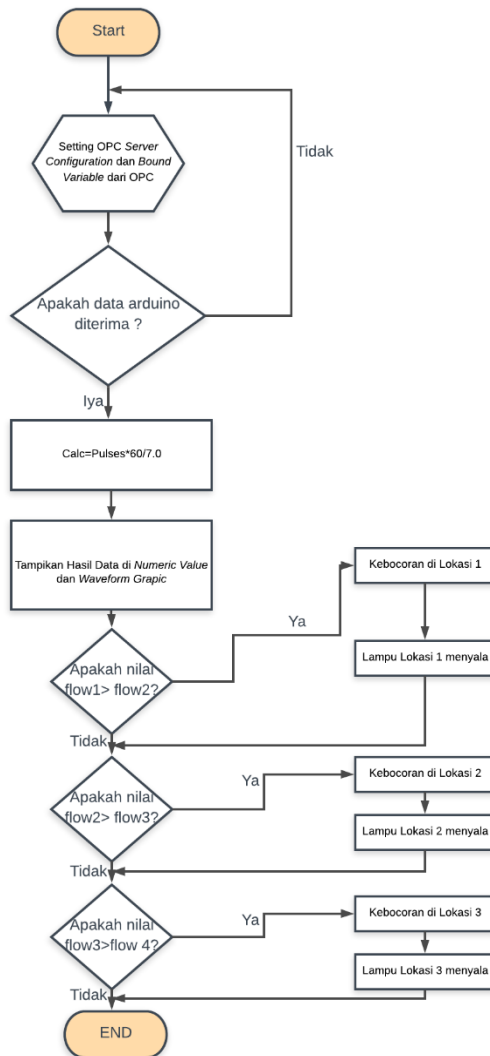
Berdasarkan **Gambar 3.6** dan **Gambar 3.7** yaitu *Flowchart* Menghubungkan setiap tombol dengan menggunakan garis penghubung. *Flowchart* merupakan langkah awal dari pembuatan program pengolahan dengan komputer, dapat dirangkum urutan dasar pemecahan suatu masalah yaitu :

- START** : instruksi untuk persiapan peralatan yang diperlukan sebelum menangani pemecahan masalah.
- READ** : instruksi untuk membaca data dari suatu peralatan input.
- PROCESS** : kegiatan yang berkaitan dengan pemecahan persoalan sesuai dengan data yang dibaca.
- WRITE** : instruksi untuk merekam hasil kegiatan ke peralatan *output*.
- END** : mengakhiri kegiatan pengolahan

Sehingga penerapan pada pemrograman sensor *water flow* menggunakan aplikasi Arduino IDE. 4 *water flow sensor* akan membaca aliran air pada 4 titik pada saluran pipa. Untuk sensor *flow* pertama diberi nama *Flow 1*, sensor kedua diberi nama *Flow 2*, sensor ketiga diberi nama *Flow 3* dan sensor *flow* terakhir diberi nama *Flow 4*. Selanjutnya dibuat program yang akan membaca debit pada 4 sensor. Kemudian program di upload ke 4 Arduino UNO yang telah disiapkan. Ketika terjadi perbedaan aliran air pada titik pipa maka akan ditampilkan oleh LabView



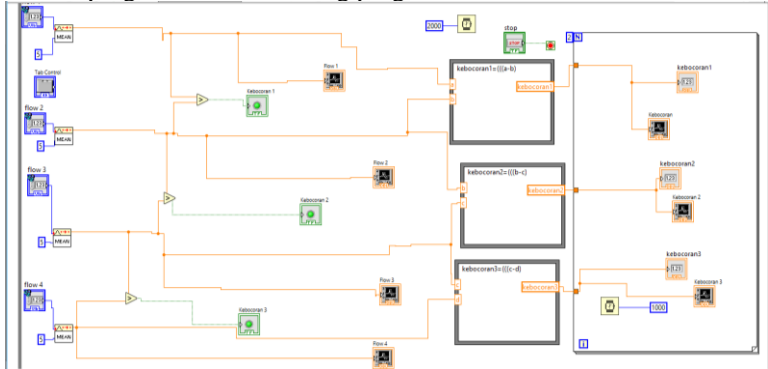
Gambar 3.6 Flowchart Program Arduino.



Gambar 3.7 *Flowchart* NIOPC

3.2.2 Perancangan Software LabView

Pada perancangan *software* LabView yang pertama dirancang yaitu membuat *block diagram* setelah itu merancang *front panel*. *Block diagram* berisi *source code* yang berfungsi sebagai instruksi untuk *front panel*. Sedangkan pada *front panel* sendiri mengandung *control* dan indikator untuk membangun sebuah VI (*Virtual Instruments*), menjalankan program dan men-debug program.



Gambar 3.8 Block Diagram LabView

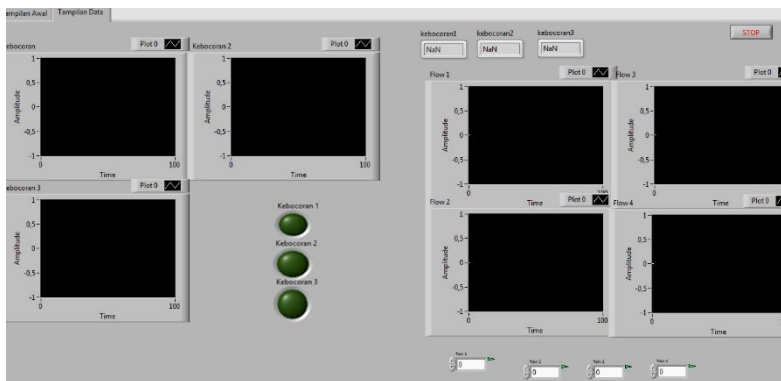
Selanjutnya yaitu membuat tampilan *front panel*. Cara memasukkan gambar pada *front panel* LabView prinsipnya tinggal *drag and drop*. Untuk mendapatkan gambar yang disediakan oleh LabView caranya adalah :

1. Pada *front panel* LabView klik menu bar “Tools”.
2. Lalu pindahkan kursor ke menu “DCS Module”.
3. Lalu klik “Image Navigator”.
4. Pada menu “Image Navigator” pilih gambar lalu *drag* dan *drop* gambar yang dibutuhkan.

Perancangan pada *block diagram* pada **Gambar 3.8** dibuat gambar instalasi pipa yang dilengkapi dengan sensor *flow* dimana pada masing-masing sensor terdapat keterangan diatasnya, selain itu terdapat *tab control* sehingga dapat mengganti *tab* pada halaman awal. Pada bagian bawah tanda nama Data1, Data 2, Data 3 dan Data 4 merupakan hasil dari pembacaan sensor.

Lalu pada bagian bawah tanda nama *Flow 1*, *Flow 2*, *Flow 3* dan *Flow 4* untuk grafik dan Nilai *Flow 1* (L/h), Nilai *Flow 2* (L/h), Nilai *Flow 3* (L/h), Nilai *Flow 4* (L/h) untuk *numeric indicator*. Nilai *Flow* langsung menghitung jumlah debit air yang masuk jadi tidak perlu ditambahkan rumus untuk menentukan debit air.

Hasil perhitungan persentase kebocoran ditampilkan pada bagian kiri. Sedangkan gambar diletakkan pada bagian kanan secara paralel, yang dimana telah terdapat tanda nama pada bagian atas gambar.



Gambar 3.9 *Front Panel LabView*

Pada **Gambar 3.9** dapat dilihat terdapat *Waveform Graph* , yang berfungsi untuk menampilkan grafik pada kebocoran di titik A , B, C dan grafik pada setiap sensor. Selain itu terdapat nilai *bit* yang dihasilkan pembacaan setiap sensor. Pada *front panel* juga terdapat lampu indikator ketika ada kebocoran di titik A, B, C dan dilengkapi juga *DSC module* yang berbentuk menyerupai pipa, sensor, dan tangki air.

BAB IV

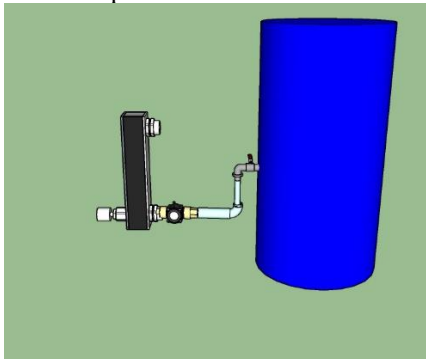
PENGUJIAN DAN ANALISA ALAT

Pada bab ini akan dibahas mengenai pengujian dan analisa data pengujian dari hasil *prototype* Rancang Bangun Pendeteksian Kebocoran Menggunakan *Water Flow Sensor* Berbasis WiFi yang telah dibuat. Pengujian *prototype* tersebut ditujukan untuk memastikan agar peralatan yang dibuat dapat berfungsi dengan baik.

Pengujian dan pengukuran pada *prototype* meliputi: pengujian *water flow* sensor, pengujian WiFi, dan pengujian *prototype* tersebut. Setelah melakukan beberapa pengujian alat, data yang diperoleh dianalisa untuk mengetahui proses kerja dari seluruh sistem alat yang dibuat.

4.1 Kalibrasi *Water Flow Sensor*

Pengujian ini bertujuan agar *water flow sensor* yang memiliki output berupa pulsa bisa linier terhadap debit dengan satuan Liter/hour. **Gambar 4.1** menunjukkan cara pengujian *water flow sensor* dengan membandingkan *water flow sensor analog* yaitu Z-3003. Produk *flow* meter tersebut dipakai di beberapa departemen di ITS terutama Departemen Teknik Kimia FTI ITS. Dalam hal ini, *flow* meter Z-3003 menghasilkan *output* berupa Liter/minute sehingga perlu dikonversikan dengan cara $LPM \times 60$, lalu dapat dibandingkan dengan pembacaan sensor *water flow* dan mencari linearitas dari sensor tersebut. Proses kalibrasi *water flow* sensor seperti **Gambar 4.1**



Gambar 4.1 Menghubungkan *Water Flow* Dengan Z-3003

Pada gambar diatas Z-3003 dihubungkan dengan menggunakan sambungan dart dalam ¾ sehingga dapat dihubungkan secara langsung ke sensor *water flow*. Kemudian,selang yang berasal dari kran dihubungkan secara langsung ke sensor *water flow* .Kemudian bandingkan hasil pembacaan sensor *water flow* dengan Z-3003,lalu hitung juga nilai eror dengan menggunakan rumus:

$$\%Error = \frac{Z3003-Flow}{Z3003} \times 100\%.....(2.3)$$

Tabel 4.1 Kalibrasi *Water Flow* Sensor 1

Z-3003 (L/m)	Z-3003 (L/h)	Flow 1 (L/h)	Pulsa	Error %
1	60	60	7	0
1 ½	90	94	11	4,4444
2	120	120	14	0
3	180	180	21	0
3 ½	210	205	24	2,38
4	240	240	28	0
5	300	300	35	0
5 ½	330	334	39	1,21
6	360	360	42	0
6 ½	390	394	46	1,02
7	420	420	49	0

Pada **Tabel 4.1** dapat dilihat hasil kalibrasi yang didapatkan oleh *flow 1* , dan eror yang terbesar yaitu 4,4444% dan nilai pulsa berbanding lurus dengan *flow* sehingga ketika nilai pulsa bertambah maka nilai *flow* juga bertambah .

Tabel 4.2 Kalibrasi *Water Flow* Sensor 2

Z-3003 (L/m)	Z-3003 (L/h)	Flow 2 (L/h)	Pulsa	Error %
1	60	60	7	0
1 ½	90	94	11	4,444
2	120	120	14	0
3	180	180	21	0

Z-3003 (L/m)	Z-3003 (L/h)	Flow 2 (L/h)	Pulsa	Error %
3 ½	210	205	24	2,38
4	240	240	28	0
5	300	300	35	0
5 ½	330	334	39	1,21
6	360	360	42	0
6 ½	390	394	46	1,02
7	420	420	49	0

Pada **Tabel 4.2** dapat dilihat hasil kalibrasi yang didapatkan oleh *flow 2* , dan eror yang terbesar yaitu 4,4444% dan nilai pulsa berbanding lurus dengan *flow* sehingga ketika nilai pulsa bertambah maka nilai *flow* juga bertambah.

Tabel 4.3 Kalibrasi *Water Flow* Sensor 3

Z-3003 (L/m)	Z-3003 (L/h)	Flow 3 (L/h)	Pulsa	Error %
1	60	60	7	0
1 ½	90	94	11	4,444
2	120	120	14	0
3	180	180	21	0
3 ½	210	214	25	1,90
4	240	240	28	0
5	300	300	35	0
5 ½	330	325	38	1,51
6	360	360	42	0
6 ½	390	394	46	1,02
7	420	420	49	0

Pada **Tabel 4.3** dapat dilihat hasil kalibrasi yang didapatkan oleh *flow 3* , dan eror yang terbesar yaitu 4,4444% dan nilai pulsa berbanding lurus dengan *flow* namun yang berbeda dengan hasil pengukuran *flow* sebelumnya adalah eror ketika pembacaan 330 L/h. Pada *flow 3* didapatkan eror yaitu 1,51%.

Tabel 4.4 Kalibrasi *Water Flow* Sensor 4

Z-3003 (L/m)	Z-3003 (L/h)	Flow 4 (L/h)	Pulsa	Eror %
1	60	60	7	0
1 ½	90	94	11	4,444
2	120	120	14	0
3	180	180	21	0
3 ½	210	214	25	1,90
4	240	240	28	0
5	300	300	35	0
5 ½	330	334	39	1,21
6	360	360	42	0
6 ½	390	394	46	1,02
7	420	420	49	0

Pada **Tabel 4.4** dapat dilihat hasil kalibrasi yang didapatkan oleh *flow* 3 , dan eror yang terbesar yaitu 4,4444% dan nilai pulsa berbanding lurus dengan *flow* namun yang berbeda dengan hasil pengukuran *flow* sebelumnya adalah eror ketika pembacaan 330 L/h yaitu 1,90%. Kesimpulan dari pengujian keempat sensor memiliki rata-rata eror yang berbeda-beda, untuk rata-rata eror keempat sensor adalah 2,1435%, 2,2635%, 2,1435% , dan 2,1435 % .

4.2 Pengukuran *Water Flow* Sensor Satu Titik

Pengujian *water flow* sensor memiliki tujuan yaitu memastikan bahwa pengukuran yang dilakukan oleh setiap sensor pada satu titik yang sama dengan harapan memiliki pembacaan sensor yang sama. Cara menguji cobanya adalah dengan cara menghubungkan pipa 30 cm ke sensor *water flow* kemudian mencatat hasil pembacaan sensor tersebut dengan berbagai kondisi , dan terakhir, menguji sensor ketiga lainnya kemudian dibandingkan.

Tabel 4.5 Pengujian *Water Flow* Sensor 1 Titik yang Sama

Keadaan Valve	Flow 1 L/h	Flow 2 L/h	Flow 3 L/h	Flow 4 L/h
50 %	634	634	625	625
100%	728	737	737	737

Pada **Gambar 4.5** dapat diberi kesimpulan dari pengujian tersebut adalah keempat *water flow* sensor membaca aliran air tersebut cenderung sama dengan keadaan tertentu dengan cara membandingkan hasil pengukurannya.

4.3 Pengujian Stop Valve

Pengujian *stop valve* memiliki tujuan yaitu menguji kemampuan pipa, dengan cara menggunakan *prototype* yang telah dibuat dan mengganti kondisi *stop valve* seperti 22,5°, 45°, dan 53°.

Tabel 4.6 Keadaan Stop Valve 22,5°

No.	Flow1 (L/h)	Flow2 (L/h)	Flow3 (L/h)	Flow 4 (L/h)
Percobaan 1	512	496	488	468
Percobaan 2	504	480	488	468
Percobaan 3	512	496	488	468

Pada **Tabel 4.6** keadaan *prototype* berjalan tanpa adanya kebocoran dan kondisi *stop valve* ditutup 22,5°, didapatkan *flow* terbesar pada *flow* 1 karena dekat dengan sumber air yaitu pompa .

Tabel 4.7 Keadaan Stop Valve 45°

No.	Flow1 (L/h)	Flow2 (L/h)	Flow3 (L/h)	Flow 4 (L/h)
Percobaan 1	536	522	520	519
Percobaan 2	536	522	522	519
Percobaan 3	536	522	520	510

Pada **Tabel 4.7** keadaan *prototype* berjalan tanpa adanya kebocoran dan kondisi *stop valve* ditutup 45°. Dapat dilihat bahwa *flow*nya naik ketika valve tersebut ditutup 45°.

Tabel 4.8 Keadaan Stop Valve 53°

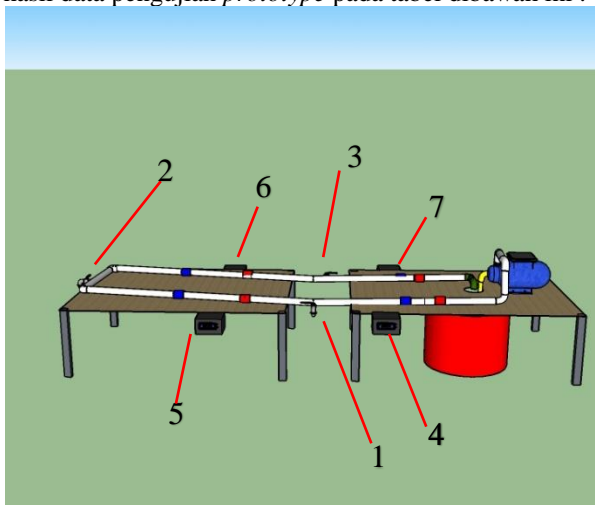
No.	Flow1 (L/h)	Flow2 (L/h)	Flow3 (L/h)	Flow 4 (L/h)
Percobaan 1	280	274	264	260
Percobaan 2	280	274	264	260
Percobaan 3	280	274	264	260

Pada **Tabel 4.8** keadaan *prototype* berjalan tanpa adanya kebocoran dan kondisi *stop valve* ditutup 53° dan dapat dilihat bahwa *flow*nya mengalami penurunan dikarenakan air yang didalam pipa tidak dapat keluar sehingga menyebabkan air mengalir kembali kedalam yang menyebabkan penurunan *flow* .

Kesimpulan dari pengujian *stop valve* adalah kemampuan pipa yang optimal adalah 53° dikarenakan ketika uji coba masih belum terjadi kebocoran di setiap sisi pipa, sedangkan jika menggunakan kondisi lebih dari itu maka pipa akan rusak .

4.4 Pengujian Kebocoran

Pengujian ini bertujuan untuk mengetahui pembacaan sensor dengan 2 kondisi yaitu kondisi normal dan kondisi kebocoran menggunakan *prototype* kami yang sudah buat seperti **Gambar 4.1**, dan tabel hasil data pengujian *prototype* pada tabel dibawah ini .



Gambar 4.2 Pengujian Menggunakan *Prototype*

1. Titik Kebocoran 1
2. Titik Kebocoran 2
3. Titik Kebocoran 3
4. Station 1
5. Station 2
6. Station 3

7. Station 4

Station merupakan kotak hitam yang didalamnya terdapat *ethernet shield* dan *Arduino UNO* yang telah terhubung dengan *water flow sensor*, yang dimana dinamakan *flow 1*, *flow 2*, *flow 3*, dan *flow 4*.

Tabel 4.9 Saat Keadaan Normal

No	Flow 1 (L/h)	Flow 2 (L/h)	Flow 3 (L/h)	Flow 4 (L/h)
Percobaan 1	512	472	496	450
Percobaan 2	512	464	504	467
Percobaan 3	512	472	496	450
Percobaan 4	512	472	496	467
Percobaan 5	512	464	496	450

Pada **Tabel 4.9** dapat dilihat keadaan *prototype* tanpa kebocoran dan keadaan *stop valve* dibuka $\pm 54^\circ$. Selain itu, setiap sensor cenderung memiliki nilai yang berbeda-beda.

Tabel 4.10 Saat Keadaan Kran Kebocoran 1 Dengan Sudut $\pm 20^\circ$

No	Flow 1 (L/h)	Flow 2 (L/h)	Flow 3 (L/h)	Flow 4 (L/h)
Percobaan 1	512	424	464	435
Percobaan 2	512	432	456	435
Percobaan 3	512	432	464	457
Percobaan 4	512	424	456	435
Percobaan 5	512	424	464	435

Pada **Tabel 4.10** dapat dilihat yang mengalami penurunan debit air adalah *flow 2*, *flow 3*, dan *flow 4* dikarenakan letaknya setelah kebocoran posisi 1, sedangkan *flow 1* tidak mengalami penurunan dikarenakan letaknya sebelum kebocoran .

Tabel 4.11 Saat Keadaan Kran Kebocoran 1 Dengan Sudut $\pm 30^\circ$

No	Flow 1 (L/h)	Flow 2 (L/h)	Flow 3 (L/h)	Flow 4 (L/h)
Percobaan 1	512	320	352	340
Percobaan 2	504	328	352	338
Percobaan 3	504	328	344	340
Percobaan 4	504	320	352	338
Percobaan 5	512	328	344	338

Perhatikan **Tabel 4.11**, dapat disimpulkan yang mengalami penurunan debit air adalah *flow 2*, *flow 3*, dan *flow 4* dikarenakan letaknya setelah kebocoran posisi 1, namun kali ini penurunannya lebih banyak dibandingkan $\pm 20^\circ$.

Tabel 4.12 Saat Keadaan Kran Kebocoran 1 Dengan Sudut $\pm 45^\circ$

No	<i>Flow 1</i> (L/h)	<i>Flow 2</i> (L/h)	<i>Flow 3</i> (L/h)	<i>Flow 4</i> (L/h)
Percobaan 1	512	232	240	236
Percobaan 2	512	240	240	240
Percobaan 3	512	240	240	236
Percobaan 4	512	232	240	236
Percobaan 5	512	232	240	240

Perhatikan **Tabel 4.12**, dapat disimpulkan yang mengalami penurunan debit air adalah *flow 2*, *flow 3*, dan *flow 4* dikarenakan letaknya setelah kebocoran posisi 1, namun kali ini penurunannya lebih banyak dibandingkan $\pm 30^\circ$.

Tabel 4.13 Saat Keadaan Kran Kebocoran 2 Dengan Sudut $\pm 20^\circ$

No	<i>Flow 1</i> (L/h)	<i>Flow 2</i> (L/h)	<i>Flow 3</i> (L/h)	<i>Flow 4</i> (L/h)
Percobaan 1	504	464	448	434
Percobaan 2	504	464	448	441
Percobaan 3	504	464	448	434
Percobaan 4	504	464	448	434
Percobaan 5	504	464	448	441

Perhatikan **Tabel 4.13**, dapat disimpulkan yang mengalami penurunan debit air adalah *flow 3*, dan *flow 4* dikarenakan letaknya setelah kebocoran posisi 2.

Tabel 4.14 Saat Keadaan Kran Kebocoran 2 Dengan Sudut $\pm 30^\circ$

No	<i>Flow 1</i> (L/h)	<i>Flow 2</i> (L/h)	<i>Flow 3</i> (L/h)	<i>Flow 4</i> (L/h)
Percobaan 1	512	464	320	300
Percobaan 2	512	464	328	310
Percobaan 3	512	472	328	300
Percobaan 4	504	464	328	310
Percobaan 5	512	464	328	300

Perhatikan **Tabel 4.14** ,dapat disimpulkan yang mengalami penurunan debit air adalah *flow* 3, dan *flow* 4 dikarenakan letaknya setelah kebocoran posisi 2.

Tabel 4.15 Saat Keadaan Kran Kebocoran 2 Dengan Sudut $\pm 45^\circ$

No	<i>Flow</i> 1 (L/h)	<i>Flow</i> 2 (L/h)	<i>Flow</i> 3 (L/h)	<i>Flow</i> 4 (L/h)
Percobaan 1	512	472	272	250
Percobaan 2	504	472	264	263
Percobaan 3	504	472	264	263
Percobaan 4	504	464	272	250
Percobaan 5	512	472	264	250

Perhatikan **Tabel 4.15**, dapat disimpulkan yang mengalami penurunan debit air adalah *flow* 3, dan *flow* 4 dikarenakan letaknya setelah kebocoran posisi 2.

Tabel 4.16 Saat Keadaan Kran Kebocoran 3 Dengan Sudut $\pm 20^\circ$

No	<i>Flow</i> 1 (L/h)	<i>Flow</i> 2 (L/h)	<i>Flow</i> 3 (L/h)	<i>Flow</i> 4 (L/h)
Percobaan 1	512	464	496	434
Percobaan 2	512	464	496	441
Percobaan 3	504	472	504	434
Percobaan 4	504	464	504	434
Percobaan 5	512	472	496	441

Perhatikan **Tabel 4.16**, dapat disimpulkan yang mengalami penurunan debit air adalah *flow* 4 dikarenakan letaknya setelah kebocoran posisi 2.

Tabel 4.17 Saat keadaan Kran Kebocoran 3 Dengan Sudut $\pm 30^\circ$

No	<i>Flow</i> 1 (L/h)	<i>Flow</i> 2 (L/h)	<i>Flow</i> 3 (L/h)	<i>Flow</i> 4 (L/h)
Percobaan 1	512	472	496	300
Percobaan 2	504	472	496	300
Percobaan 3	504	472	496	300
Percobaan 4	504	464	496	300
Percobaan 5	512	472	496	300

Perhatikan **Tabel 4.17** ,dapat disimpulkan yang mengalami penurunan debit air adalah *flow* 4 dikarenakan letaknya setelah kebocoran posisi 2 namun penurunanya lebih banyak dibandingkan dengan posisi 20°.

Tabel 4.18 Saat Keadaan Kran Kebocoran 3 Dengan Sudut $\pm 45^\circ$

No	<i>Flow</i> 1 (L/h)	<i>Flow</i> 2 (L/h)	<i>Flow</i> 3 (L/h)	<i>Flow</i> 4 (L/h)
Percobaan 1	512	472	496	245
Percobaan 2	504	472	504	269
Percobaan 3	504	472	496	269
Percobaan 4	504	464	496	269
Percobaan 5	512	472	496	245

Perhatikan **Tabel 4.18**, dapat disimpulkan yang mengalami penurunan debit air adalah *flow* 4 dikarenakan letaknya setelah kebocoran posisi 2 namun penurunanya lebih banyak dibandingkan dengan posisi 30°. Kesimpulan dari pengujian tersbeut adalah ketika terjadi kebocoran di titik 1 maka akan memperubah nilai *flow* pada sensor *flow* 1, *flow* 2, *flow* 3 dan *flow* 4. Sedangkan jika letak kebocoran berada ditengah-tengah sensor *water flow* seperti kebocoran di titik B yaitu berada di posisi antara *flow* 2, *flow* 3, *flow* 4 maka nilai *flow* 1 tidak berubah namun yang mengalami perubahan adalah *flow* 3 dan *flow* 4 .

4.5 Pengujian Koneksi *Ethernet*

Pengujian ini dilakukan pada modul *Ethernet shield* yang telah dipasang pada bagian atas *board* Arduino UNO. Dalam hal ini pengujian komunikasi dibagi menjadi dua, yaitu pengujian komunikasi *Ethernet* pada *board* Arduino UNO dan pengujian komunikasi *Ethernet* dengan *software LabVIEW*.

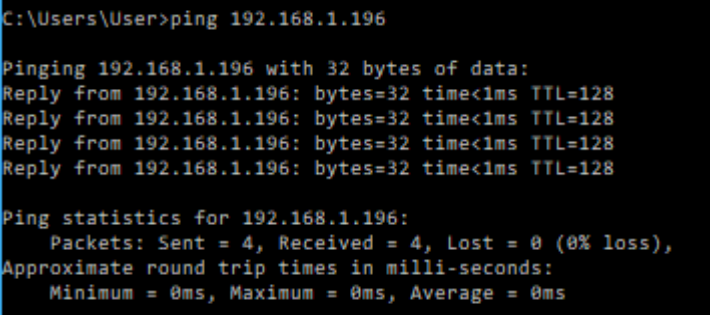
4.5.1 Pengujian pada *Board* Arduino UNO

Sebelum menggunakan komunikasi *Ethernet*, sebaiknya dilakukan pengujian komunikasi *Ethernet* pada *board* Arduino UNO terlebih dahulu. Pengujian ini dilakukan dengan cara menyamakan *setting* IP (*Internet Protocol*) *address* yang telah diprogram pada *board* Arduino UNO, dapat dilihat pada **Gambar 4.3** , **Gambar 4.4** , **Gambar 4.5**, **Gambar 4.6**, **Gambar 4.7**, **Gambar 4.8**, **Gambar 4.9**, dan **Gambar 4.10** yang bergaris bawah merah, dengan *setting* IP *address* pada *ethernet*

shield. Setting IP address pada *ethernet shield* dapat diketahui melalui *command prompt* pada komputer.

```
void setup()
{
    lcd.begin(16, 2);
    byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
    byte ip[] = { 192, 168, 1, 196 };
    mb.config(mac, ip);
    pinMode(psensor, INPUT);
    Serial.begin(9600);
    attachInterrupt(0, rpm, RISING);
    mb.addIreg(sencalc);
    mb.addIreg(water);
    sei();
}
```

Gambar 4.3 Program Pada Arduino UNO ke-1



```
C:\Users\User>ping 192.168.1.196

Pinging 192.168.1.196 with 32 bytes of data:
Reply from 192.168.1.196: bytes=32 time<1ms TTL=128
Reply from 192.168.1.196: bytes=32 time<1ms TTL=128
Reply from 192.168.1.196: bytes=32 time<1ms TTL=128
Reply from 192.168.1.196: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.196:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

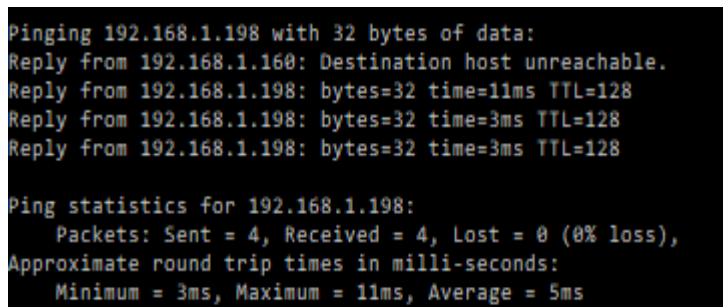
Gambar 4.4 *Command Prompt* Pada Komputer Dengan IP Address Arduino UNO ke-1

```

void setup()
{
    lcd.begin(16, 2);
    byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
    byte ip[] = { 192, 168, 1, 198 };
    mb.config(mac, ip);
    pinMode(psensor, INPUT);
    Serial.begin(9600);
    attachInterrupt(0, rpm, RISING);
    mb.addIreg(sencalc);
    mb.addIreg(water);
    sei();
}

```

Gambar 4.5 Program Pada Arduino UNO ke-2



```

Pinging 192.168.1.198 with 32 bytes of data:
Reply from 192.168.1.160: Destination host unreachable.
Reply from 192.168.1.198: bytes=32 time=11ms TTL=128
Reply from 192.168.1.198: bytes=32 time=3ms TTL=128
Reply from 192.168.1.198: bytes=32 time=3ms TTL=128

Ping statistics for 192.168.1.198:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 11ms, Average = 5ms

```

Gambar 4.6 *Command Prompt* pada Arduino 2

```

void setup()
{
    lcd.begin(16, 2);
    byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
    byte ip[] = { 192, 168, 1, 200 };
    mb.config(mac, ip);
    pinMode(psensor, INPUT);
    Serial.begin(9600);
    attachInterrupt(0, rpm, RISING);
    mb.addIreg(sencalc);
    mb.addIreg(water);
    sei();
}

```

Gambar 4.7 Program Arduino ke- 3

```
C:\Users\User>ping 192.168.1.200

Pinging 192.168.1.200 with 32 bytes of data:
Reply from 192.168.1.200: bytes=32 time=5ms TTL=128
Reply from 192.168.1.200: bytes=32 time=1ms TTL=128
Reply from 192.168.1.200: bytes=32 time=3ms TTL=128
Reply from 192.168.1.200: bytes=32 time=2ms TTL=128

Ping statistics for 192.168.1.200:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 5ms, Average = 2ms
```

Gambar 4.8 Command Prompt Arduino ke-3

```
void rpm () //This is the function that the interrupt calls
{
    NbTopsFan++; //This function measures the rising and falling edge of the hall effect sensors signal
}

ModbusIP mb;

long ts;
// The setup() method runs once, when the sketch starts
void setup() //
{
    // The media access control (ethernet hardware) address for the shield
    byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEF };
    // The IP address for the shield
    byte ip[] = { 192, 168, 0, 202 };
    //config Modbus IP
    mb.config(mac, ip);

    pinMode(hallsensor, INPUT); //initializes digital pin 2 as an input
    Serial.begin(9600); //This is the setup function where the serial port is initialised,
    attachInterrupt(0, rpm, RISING); //and the interrupt is attached

    mb.addIreg(sencalc);
    NbTopsFan = 0; //Set NbTops to 0 ready for calculations
    sei();
}
```

Gambar 4.9 Program Arduino ke-4

```
C:\Users\Laptop Ku>ping 192.168.0.202

Pinging 192.168.0.202 with 32 bytes of data:
Reply from 192.168.0.202: bytes=32 time=1ms TTL=128
Reply from 192.168.0.202: bytes=32 time=2ms TTL=128
Reply from 192.168.0.202: bytes=32 time=2ms TTL=128
Reply from 192.168.0.202: bytes=32 time=2ms TTL=128

Ping statistics for 192.168.0.202:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms

C:\Users\Laptop Ku>ping 192.168.0.202

Pinging 192.168.0.202 with 32 bytes of data:
Reply from 192.168.0.202: bytes=32 time=1ms TTL=128
Reply from 192.168.0.202: bytes=32 time=2ms TTL=128
Reply from 192.168.0.202: bytes=32 time=2ms TTL=128
Reply from 192.168.0.202: bytes=32 time=2ms TTL=128

Ping statistics for 192.168.0.202:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms

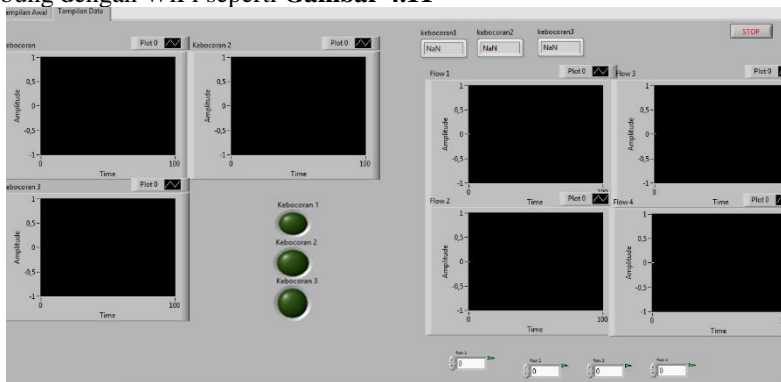
C:\Users\Laptop Ku>
```

Gambar 4.10 Command Prompt Arduino ke-4

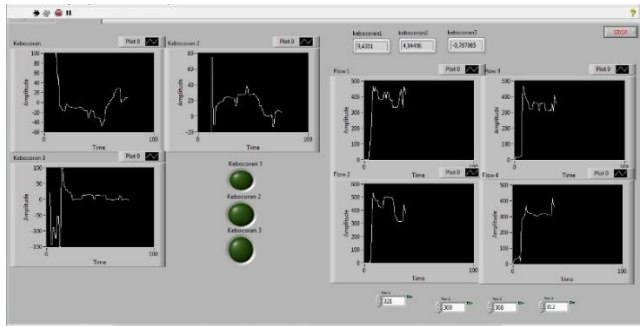
Dapat dilihat pada **Gambar 4.3** , **Gambar 4.4** , **Gambar 4.5**, **Gambar 4.6**, **Gambar 4.7**, **Gambar 4.8**, **Gambar 4.9**, dan **Gambar 4.10**. menunjukkan bahwa *ethernet shield* yang tersambung pada komputer telah siap digunakan. Ketika mengirim perintah *ping* nomor IP *address* pada kolom *command prompt* maka komputer akan memunculkan informasi bahwa nomor IP *address* telah bekerja.

4.6 Pengujian Data *Telemetry*

Pengujian secara *telemetry* dilakukan dengan cara menggunakan *prototype* yang memiliki 4 sensor dengan 3 titik kebocoran yang nantinya akan diuji coba dengan beberapa kondisi. Hal yang pertama-tama harus dilakukan adalah menghubungkan Arduino UNO, *ethernet shield* dan *water flow* sensor meter dengan cara menghubungkan Arduino 1 dengan kabel RJ-45, Arduino 2 dengan *acces point*, Arduino 3 dihubungkan dengan *acces point*, dan yang terakhir adalah Arduino 4 dihubungkan dengan *acces point*. Setelah itu buka tampilan LabView dan pilih tampilan data, kemudian cek kondisi awal tampilan LabView yang sudah terhubung dengan WiFi seperti **Gambar 4.11**



Gambar 4.11 Tampilan Awal *LabView* Terhubung Dengan Wi-Fi



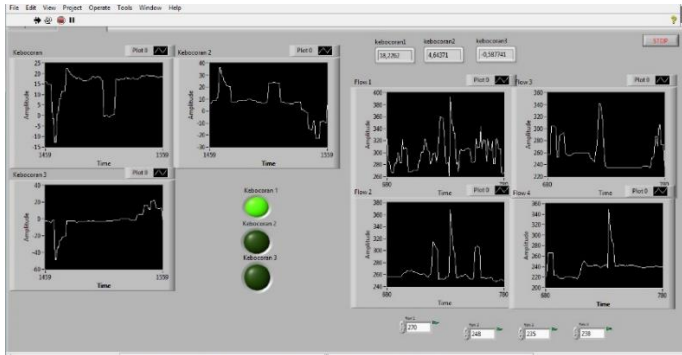
Gambar 4.12 Tampilan Kondisi Normal Tanpa Kebocoran

Pada **Gambar 4.12** dapat dilihat pada uji coba *prototype* dengan keadaan normal dan tanpa titik kebocoran kemudian keadaan *stop valve* $\pm 53^\circ$, didapati nilai *flow* 1 terbaca 321 L/h, *flow* 2 terbaca 300 L/h, *flow* 3 terbaca 308 L/h, dan *flow* 4 312 L/h. Lalu dapat dilihat juga presentase kebocorannya adalah 9 % di titik 1, 4% di titik 2 dan 0% di titik 3 .



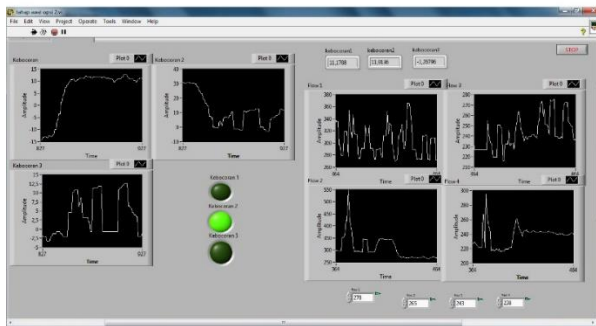
Gambar 4.13 Tampilan Kondisi Bocor $\pm 20^\circ$ di titik 1.

Pada **Gambar 4.13** alat *prototype* dijalankan dengan kondisi kebocoran $\pm 20^\circ$ di titik 1, dapat dilihat nilai *flow* 1 270 L/h, *flow* 2 257 L/h, *flow* 3 252 L/h, *flow* 4 246 L/h. Sedangkan untuk nilai kebocoran 1 meningkat menjadi 10 %, sedangkan kebocoran 2 menjadi 5%, sedang kebocoran 3 tetap bernilai 0% .Kemudian dapat diamati bahwa lampu di lokasi kebocoran 1 menyala, dan dilihat juga pada grafik setiap *flow* yang menandakan bahwa nilainya berubah-ubah.



Gambar 4.14 Tampilan Kondisi Bocor $\pm 30^\circ$ di Titik 1.

Pada **Gambar 4.14** dapat dilihat nilai *flow* 1 bernilai 270 L/h, *flow* 2 bernilai 248 L/h, *flow* 3 bernilai 235 L/h, dan *flow* 4 bernilai 235 L/h sedangkan presentase kebocoran 1 bernilai 18% ,lalu kebocoran 2 bernilai 4% dan yang terakhir kebocoran 0%. Kemudian dapat diamati bahwa lampu di lokasi kebocoran 1 menyala, dan perhatikan grafik *flow* serta presentase kebocoran, jika dibandingkan dengan $\pm 20^\circ$ grafik yang dimiliki cenderung berbeda karena memiliki amplitudo yang tinggi, dan nilai presentase kebocorannya lebih besar.



Gambar 4.15 Tampilan Kondisi Bocor $\pm 20^\circ$ di Titik 2

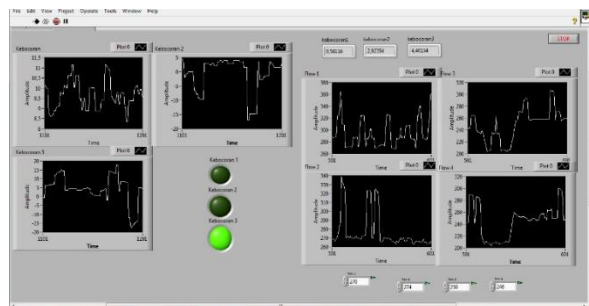
Pada **Gambar 4.15** dapat dilihat nilai *flow* 1 bernilai 278 L/h, *flow* 2 bernilai 265 L/h, *flow* 3 bernilai 243 L/h, dan *flow* 4 bernilai 238 L/h sedangkan presentase kebocoran 1 bernilai 11,7%, lalu kebocoran 2 bernilai 11,91% dan yang terakhir kebocoran 3 1%. Kemudian dapat diamati bahwa lampu di lokasi kebocoran 2 menyala, dan perhatikan

grafik *flow* serta grafik presentase kebocoran, jika diamati grafiknya cenderung berbeda karena memiliki amplitudo yang tinggi.



Gambar 4.16 Tampilan Kondisi Bocor $\pm 30^\circ$ di Titik 2

Pada **Gambar 4.16** dapat dilihat nilai *flow* 1 bernilai 278 L/h, *flow* 2 bernilai 274 L/h, *flow* 3 bernilai 348 L/h, dan *flow* 4 bernilai 205 L/h sedangkan presentase kebocoran 1 bernilai 9,89% ,lalu kebocoran 2 bernilai 13,29% dan yang terakhir kebocoran 3 13,1%. Kemudian dapat diamati bahwa lampu di lokasi kebocoran 2 menyala, dan perhatikan grafik *flow* serta presentase kebocoran, jika dibandingkan dengan $\pm 20^\circ$ grafik yang dimiliki cenderung berbeda karena memiliki amplitudo yang tinggi, dan nilai presentase kebocorannya lebih besar.



Gambar 4.17 Tampilan Kondisi Bocor $\pm 20^\circ$ di Titik 3

Pada **Gambar 4.17** dapat dilihat nilai *flow* 1 bernilai 274 L/h, *flow* 2 bernilai 274 L/h, *flow* 3 bernilai 259 L/h, dan *flow* 4 bernilai 246 L/h sedangkan presentase kebocoran 1 bernilai 9,56% ,lalu kebocoran 2 bernilai 2,023% dan yang terakhir kebocoran 3 4,401% . Kemudian dapat diamati bahwa lampu di lokasi kebocoran 3 menyala, dan perhatikan grafik *flow* serta grafik presentase kebocoran, jika diamati grafiknya cenderung berbeda karena memiliki amplitudo yang tinggi.



Gambar 4.18 Tampilan Kondisi Bocor $\pm 30^\circ$ di Titik 3

Pada **Gambar 4.18** dapat dilihat nilai *flow* 1 bernilai 270 L/h, *flow* 2 bernilai 257 L/h, *flow* 3 bernilai 259 L/h, dan *flow* 4 bernilai 213 L/h sedangkan presentase kebocoran 1 bernilai 17% ,lalu kebocoran 2 bernilai -17,98% dan yang terakhir kebocoran 3 26,02%. Kemudian dapat diamati bahwa lampu di lokasi kebocoran 1 menyala, dan perhatikan grafik *flow* serta presentase kebocoran, jika dibandingkan dengan $\pm 20^\circ$ grafik yang dimiliki cenderung berbeda karena memiliki amplitudo yang tinggi, dan nilai presentase kebocorannya lebih besar.

BAB V

PENUTUP

Setelah melakukan perancangan dan pembuatan alat serta pengujian dan analisa, maka dapat ditarik kesimpulan dan saran dari kegiatan yang telah dilakukan untuk pengembangan Tugas Akhir ini.

5.1 Kesimpulan

Dari seluruh tahapan yang sudah dilaksanakan pada penyusunan Tugas Akhir ini, mulai dari studi *literature*, perancangan dan pembuatan sampai pada pengujiannya maka dapat disimpulkan bahwa:

- Nilai kebocoran yang mampu dideteksi adalah 0-18% .
- Nilai rata-rata setiap sensor adalah 2,1735% dan eror yang paling besar adalah 4,4% .
- Ketika terjadi kebocoran maka nilai sensor yang letaknya berada disetelah kebocoran nilainya akan turun, sedangkan sensor yang berada di sebelum kebocoran penurunannya tidak banyak
- Setiap sensor dengan tipe yang sama memiliki sensitifitas yang berbeda-beda sehingga harus mengganti rumus di program agar mendekati nilai yang sama .
- Konstruksi *prototype* mempengaruhi hasil pembacaan sensor *water flow*.

5.2 Saran

Untuk lebih memperbaiki dan menyempurnakan kinerja dari alat ini, maka perlu disarankan :

1. Memakai sensor tipe yang memiliki kemampuan membaca debit yang lebih besar dan memiliki sensitifitas yang sama
2. Menyesuaikan tekanan pompa dengan *waterflow* sensor yang anda pakai.
3. Memakai RTC dan *SD Card* sehingga proses pengambilan data lebih mudah .
4. Konstruksi *prototype* dibuat dengan bahan pipa yang lebih baik lagi dan jangan memakai sambungan (*water mur*) pada pipa karena dapat mengurangi aliran air di setiap titik .
5. Perlu ditambahkan fitur untuk memberikan peringatan ketika ada kebocoran bisa berupa SMS .

-----Halaman ini sengaja dikosongkan-----

DAFTAR PUSTAKA

- [1] Jajeli,Rois,(2017),”**Pipa PDAM Bocor, Tiga Ribu Warga Surabaya Kesulitan Pasokan Air**”, <https://news.detik.com/berita-jawa-timur/d-3386209/pipa-pdam-bocor-tiga-ribu-warga-surabaya-kesulitan-pasokan-air>[10 Mei 2018]
- [2] Haryanto,Duwi,”Perancangan Alat Deteksi Letak Kebocoran Pipa PVC Menggunakan *Flow Meter* Model FS300A Berbasis TCP/IP”.**Tugas Akhir**,Fisika FMIPA Universita Lampung.2016.
- [3] Santoso, B., Indarto, Deendarlianto, dan T. S. Widodo, “*Deteksi Kebocoran Pipa Pada Aliran Dua Fase Plug Menggunakan Analisis Fluktuasi Beda Tekanan*”.**Jurnal Energi dan Manufaktur. Vol. 6 No.1. 1-8 hlm.**2013
- [4] Haliday,David,**Fisika Edisi ke 3 Jilid 1** , Erlangga , Jakarta,1987
- [5] Giancolli,Douglas,**Fisika Edisi ke 5 Jilid 1**, Erlangga , Jakarta, 2001
- [6] Fadilah,Farah,”Telemetry Kebocoran Pada Pipa Distribusi Air Dengan Komunikasi Ethernet”.**Tugas Akhir**,Departemen Teknik Elektro Otomasi Fakultas Vokasi .2018.
- [7] Monk,Simon,”**Electronic Cookbook : Practical Engineering with Arduino and Raspberry Pi**”,O’Reilly Media, Virginia,2017.
- [8] Halvorsen,Hans Peter ,”**OPC and Real-Time Systems in LabView**”, Norwegia,2011.
- [9] Djuandi,Feri,**Pengenalan Arduino**, Elexmedia, Jakarta, 2011
- [10] Dharmasetiawan, Martin,**Sistem Perpipaan Distribusi Air Minum**, Ekamitra Engineering, Jakarta, 2004

LAMPIRAN A

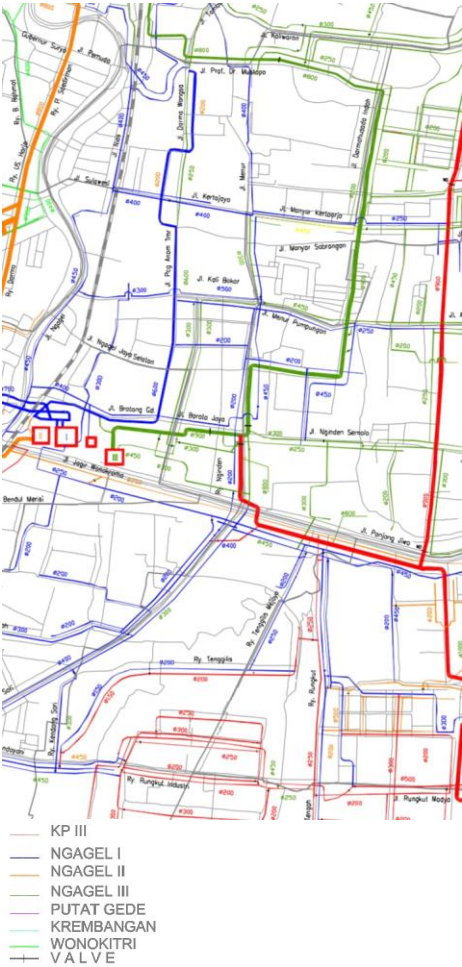
1. Gambar *Prototype* Tampak Depan



2. Gambar *Prototype* Tampak Samping



4. Gambar Distribusi Air di Surabaya



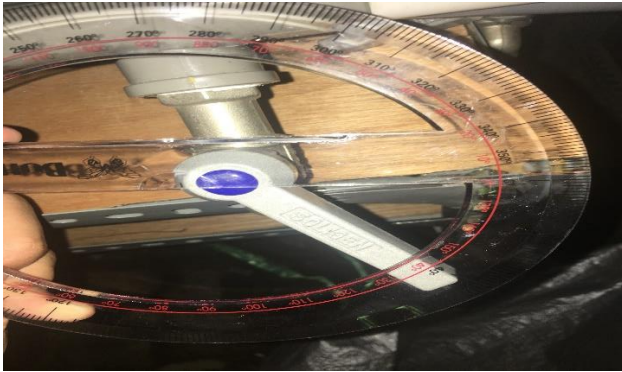
5. Dokumentasi Kran Kebocoran Saat Dibuka $\pm 20^\circ$



6. Dokumentasi Kran Kebocoran Saat Dibuka $\pm 30^\circ$



7. Dokumentasi Kran Kebocoran Saat Dibuka $\pm 45^\circ$



8. Dokumentasi *Stop valve* Saat Ditutup $\pm 22,5^{\circ}$



9. Dokumentasi *Stop valve* Saat Ditutup $\pm 45^{\circ}$



10. Dokumentasi *Stop valve* Saat Ditutup $\pm 53^\circ$



LAMPIRAN B

1. Program Arduino 1

```
#include <LiquidCrystal.h>
#include <SPI.h>
#include <Ethernet.h>
#include <Modbus.h>
#include <ModbusIP.h>

const int sencalc = 100;
const int water = 150;

double press_psi;
long ts;
LiquidCrystal lcd(9, 8, 7, 6, 5, 3);
volatile int NbTopsFan;
int Calc;
int psensor = 2;
ModbusIP mb;

void rpm ()
{
    NbTopsFan++;
}

void setup()
{
    lcd.begin(16, 2);
    byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE,
0xED };
    byte ip[] = { 192, 168, 1, 198 };
    mb.config(mac, ip);
    pinMode(psensor, INPUT);
    Serial.begin(9600);
    attachInterrupt(0, rpm, RISING);
    mb.addIreg(sencalc);
    mb.addIreg(water);
    sei();
}
```

```

}

void loop ()
{
    mb.task();
    if (millis() > ts + 1000) {
        ts = millis();
        cli();

        int val=analogRead(A0);
        press_psi = (((val*0.015)+1.06)*14.5-
27);
        int sensorVal = val;
        mb.Ireg(sencalc, sensorVal);
        Serial.print (press_psi);
        Serial.println (" Psi");
        lcd.setCursor(0,0);
        lcd.print("Psi : ");
        lcd.print(press_psi);

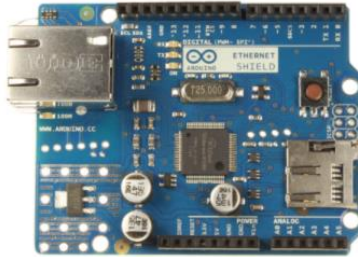
        Calc = (NbTopsFan *60 / 7.0);
        mb.Ireg(water, Calc);
        Serial.print (NbTopsFan);
        Serial.println (" L/hours");
        lcd.setCursor(0,1);
        lcd.print("Flow : ");
        lcd.print(Calc);
        lcd.print(" L/hours");
        NbTopsFan = 0;
        delay(500);
        sei();
    }
}

```

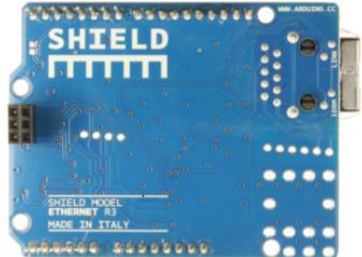
LAMPIRAN C

1. *Datasheet Ethernet shield*

Arduino Ethernet Shield V1



Arduino Ethernet Shield R3 Front



Arduino Ethernet Shield R3 Back

Overview

NOTE: this product is currently retired and the documentation will not be kept up-to-date

The Arduino Ethernet Shield V1 connects your Arduino to the internet in mere minutes. Just plug this module onto your Arduino board, connect it to your network with an RJ45 cable (not included) and follow a few simple instructions to start controlling your world through the internet. As always with Arduino, every element of the platform – hardware, software and documentation – is freely available and open-source. This means you can learn exactly how it's made and use its design as the starting point for your own circuits. Hundreds of thousands of Arduino boards are already fueling people's creativity all over the world, everyday. Join us now, Arduino is you!

- Requires an Arduino board (not included)
- Operating voltage 5V (supplied from the Arduino Board)
- Ethernet Controller: W5100 with internal 16K buffer
- Connection speed: 10/100Mb
- Connection with Arduino on SPI port

Description

The Arduino Ethernet Shield V1 allows an Arduino board to connect to the internet. It is based on the [Wiznet W5100](#) ethernet chip ([datasheet](#)). The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. Use the [Ethernet library](#) to write sketches which connect to the internet using the shield. The ethernet shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top.

The most recent revision of the board exposes the I/O pinout on rev 3 of the Arduino UNO board.

The Ethernet Shield V1 has a standard RJ-45 connection, with an integrated line transformer and Power over Ethernet enabled.

There is an onboard micro-SD card slot, which can be used to store files for serving over the network. It is compatible with all the Arduino/Genuino boards. The on-board micro SD card reader is accessible through the SD Library. When working with this library, SS is on Pin 4. The original revision of the shield contained a full-size SD card slot; this is not supported.

The shield also includes a reset controller, to ensure that the W5100 Ethernet module is properly reset on power-up. Previous revisions of the shield were not compatible with the Mega and need to be manually reset after power-up.

Download: [arduino-ethernet-shield-06-schematic.pdf](#), [arduino-ethernet-shield-06-reference-design.zip](#)

The current shield has a Power over Ethernet (PoE) module designed to extract power from a conventional twisted pair Category 5 Ethernet cable:

- IEEE802.3af compliant
- Low output ripple and noise (100mVpp)
- Input voltage range 36V to 57V
- Overload and short-circuit protection
- 9V Output
- High efficiency DC/DC converter: typ 75% @ 50% load
- 1500V isolation (input to output)

NB: the Power over Ethernet module is proprietary hardware not made by Arduino, it is a third party accessory. For more information, see the [datasheet](#)

The shield does not come with the PoE module built in, it is a separate component that must be added on.

Arduino communicates with both the W5100 and SD card using the SPI bus (through the ICSP header). This is on digital pins 10, 11, 12, and 13 on the Uno and pins 50, 51, and 52 on the Mega. On both boards, pin 10 is used to select the W5100 and pin 4 for the SD card. These pins cannot be used for general I/O. On the Mega, the hardware SS pin, 53, is not used to select either the W5100 or the SD card, but it must be kept as an output or the SPI interface won't work.

Note that because the W5100 and SD card share the SPI bus, only one can be active at a time. If you are using both peripherals in your program, this should be taken care of by the corresponding libraries. If you're not using one of the peripherals in your program, however, you'll need to explicitly deselect it. To do this with the SD card, set pin 4 as an output and write a high to it. For the W5100, set digital pin 10 as a high output.

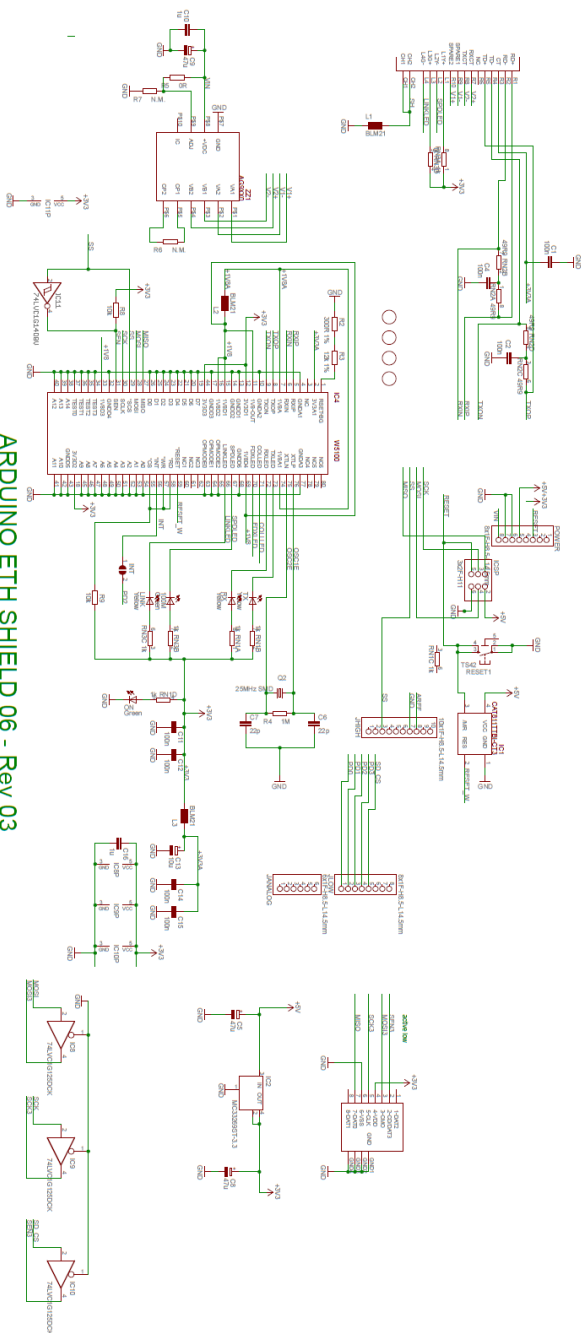
The shield provides a standard RJ45 ethernet jack.

The reset button on the shield resets both the W5100 and the Arduino board.

The shield contains a number of informational LEDs:

- PWR: indicates that the board and shield are powered
- LINK: indicates the presence of a network link and flashes when the shield transmits or receives data
- FULLD: indicates that the network connection is full duplex
- 100M: indicates the presence of a 100 Mb/s network connection (as opposed to 10 Mb/s)
- RX: flashes when the shield receives data
- TX: flashes when the shield sends data
- COLL: flashes when network collisions are detected

The solder jumper marked "INT" can be connected to allow the Arduino board to receive interrupt-driven notification of events from the W5100, but this is not supported by the Ethernet library. The jumper connects the INT pin of the W5100 to digital pin 2 of the Arduino.



REFERENCE DESIGNS ARE PROVIDED "AS IS" WITH ALL FAULTS. ARDUINO DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. ARDUINO MAKES NO REPRESENTATION OR WARRANTY ABOUT THE ACCURACY, COMPLETENESS, OR USEFULNESS OF THE INFORMATION PROVIDED HEREIN. ARDUINO WILL NOT BE RESPONSIBLE FOR DAMAGES OF ANY KIND, INCLUDING DIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, ARISING OUT OF, OR IN CONNECTION WITH, THE USE OF THE INFORMATION CONTAINED HEREIN. THESE FURTHER DEFINITION AND SHALL HAVE NO LIABILITY WHATSOEVER FOR CONFLICTS OR INCOMPATIBILITIES ARISING FROM FUTURE CHANGES TO THEM. THE PRODUCT INFORMATION ON THE WEB SITE OR MATERIALS IS SUBJECT TO CHANGE WITHOUT NOTICE. DO NOT FINALIZE A DESIGN WITH THIS INFORMATION.

ARDUINO IS A REGISTERED TRADEMARK.

W5100 Datasheet

The W5100 is a full-featured, single-chip Internet-enabled 10/100 Ethernet controller designed for embedded applications where ease of integration, stability, performance, area and system cost control are required. The W5100 has been designed to facilitate easy implementation of Internet connectivity without OS. The W5100 is IEEE 802.3 10BASE-T and 802.3u 100BASE-TX compliant.

The W5100 includes fully hardwired, market-proven TCP/IP stack and integrated Ethernet MAC & PHY. Hardwired TCP/IP stack supports TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE which has been proven in various applications for several years. 16Kbytes internal buffer is included for data transmission. No need of consideration for handling Ethernet Controller, but simple socket programming is required.

For easy integration, three different interfaces like memory access way, called direct, indirect bus and SPI, are supported on the MCU side.

Target Applications

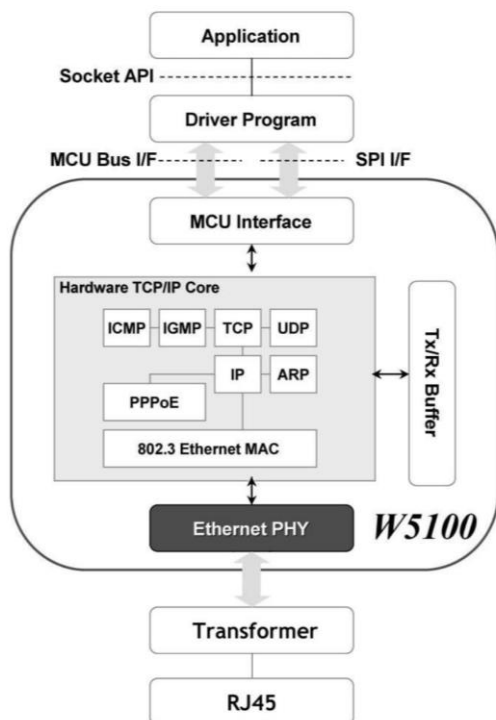
The W5100 is well suited for many embedded applications, including:

- Home Network Devices: Set-Top Boxes, PVRs, Digital Media Adapters
- Serial-to-Ethernet: Access Controls, LED displays, Wireless AP relays, etc.
- Parallel-to-Ethernet: POS / Mini Printers, Copiers
- USB-to-Ethernet: Storage Devices, Network Printers
- GPIO-to-Ethernet: Home Network Sensors
- Security Systems: DVRs, Network Cameras, Kiosks
- Factory and Building Automations
- Medical Monitoring Equipments
- Embedded Servers

Features

- Support Hardwired TCP/IP Protocols : TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE, Ethernet
- 10BaseT/100BaseTX Ethernet PHY embedded
- Support Auto Negotiation (Full-duplex and half duplex)
- Support Auto MDI/MDIX
- Support ADSL connection (with support PPPoE Protocol with PAP/CHAP Authentication mode)
- Supports 4 independent sockets simultaneously
- Not support IP Fragmentation
- Internal 16Kbytes Memory for Tx/Rx Buffers
- 0.18 μ m CMOS technology
- 3.3V operation with 5V I/O signal tolerance
- Small 80 Pin LQFP Package
- Lead-Free Package
- Support Serial Peripheral Interface(SPI MODE 0, 3)
- Multi-function LED outputs (TX, RX, Full/Half duplex, Collision, Link, Speed)

Block Diagram





10. Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

10.1 8209D – 11/10

1. Updated footnote 1 in "Features" on page 1.
2. Removed the chapter "Disclaimer" from the datasheet.
3. Updated the table [Table 27-18 on page 305](#) with a correct reference for Read Fuse bits.
4. Updated "SPI Serial Programming Characteristics" on page 306 with correct link.
5. Added typical values for R_{in} and C_{in} (both "single ended input" and "differential inputs") in [Table 25-7 on page 314](#).
6. Added "PCICR" in "Register Summary" on page 320.
7. Editing updates.
8. Updated the last page according to Atmel new Brand Style Guide.

10.2 8209C – 05/10

1. Replaced 32M1-A package information drawing with PV drawing on page 334.
2. Updated ordering information with correct info on PV package.

10.3 8209B – 10/09

1. Updated "Temperature Measurement" on page 236.
2. Updated "Manufacturing Calibration" on page 237.

10.4 8209A – 08/09

1. Initial revision.

Table of Contents

	Features	1
1	Pin Configurations	2
1.1	Pin Descriptions	3
2	Overview	5
2.1	Block Diagram	6
2.2	Pin Descriptions	7
3	Resources	9
4	About Code Examples	9
5	Data Retention	9
6	Register Summary	10
7	Errata	14
7.1	Errata ATmega16M1	14
7.2	Errata ATmega32M1	14
7.3	Errata ATmega64M1	14
8	Ordering Information	15
8.1	ATmega16M1	15
8.2	ATmega32M1	16
8.3	ATmega64M1	17
9	Packaging Information	18
9.1	32A	18
9.2	PV	19
10	Datasheet Revision History	20
10.1	8209D – 11/10	20
10.2	8209C – 05/10	20
10.3	8209B – 10/09	20
10.4	8209A – 08/09	20
	Table of Contents	i





Atmel Corporation
2305 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-0600
www.atmel.com

Atmel Asia Limited
Unit 1-G & 16, 15/F
SEA Tower, Millennium City 5
418 Kowloon Tong Road
Kowloon, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan
9F, Tonebu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0003
JAPAN
Tel: (+81)(3) 3523-3551
Fax: (+81)(3) 3523-7581

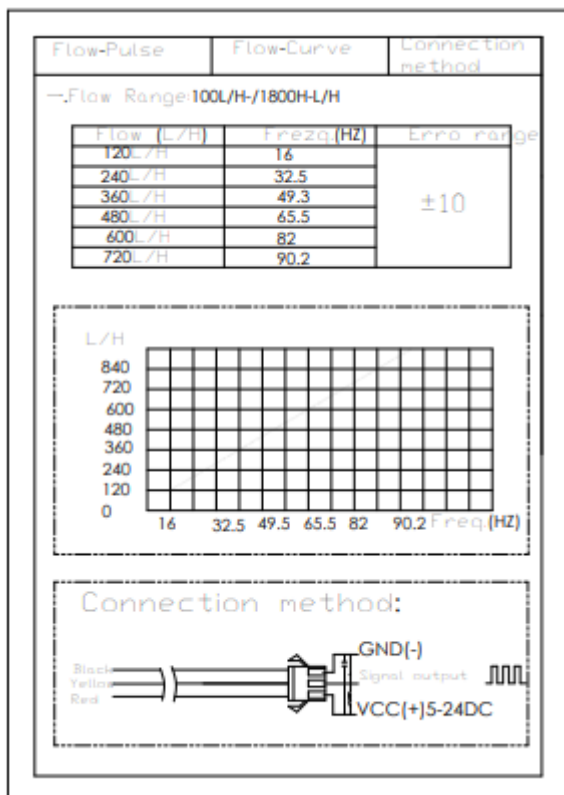
© 2010 Atmel Corporation. All rights reserved. / Rev. CORP072610

Atmel®, logo and combinations thereof, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Disclaimer: The information in this document is provided in connection with Atmel products. Its license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products, EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE. ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein, unless specifically provided otherwise. Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

820626-NVR-11/10

2. Datasheet Water Flow Sensor YF-S201



YIFA the plastics Ltd Product Introduction

1.Model:YF-21

2.Product Name:Full sensor

3.Flow Range: 1-30L/MIN

4.[1]Connection Method



(2)Voltage Range: 3.5-24VDC, Pulse Characteristic: $F=7Q(L/MIN)$

(3)Extent of error:±5%

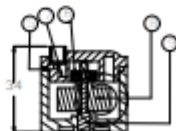
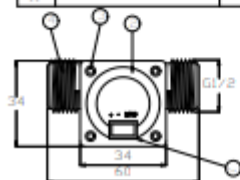
(4)Flow-Pulse

2L/MIN=16HZ 4L/MIN=32.5HZ 6L/MIN=49.3HZ

8L/MIN=65.5HZ 10L/MIN=82HZ

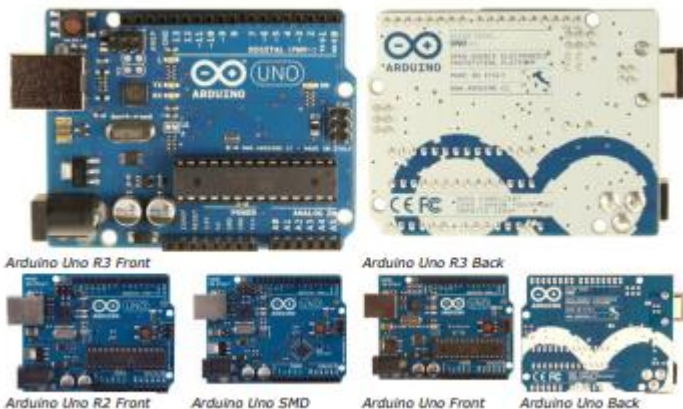
5.Bom

No.	Item	Material	Qty.
1	Connection wire		1
2	Bonnet	PA	1
3	Screw		4
4	Valve body	PA	1
5	Leak press valve		1
6	Magport		1
7	Ball		1
8	Impeller	POM	1
9	Rustless steel axis	SUS304	1
10			
11			



3.Datasheet Arduino UNO

Arduino Uno



Overview

The Arduino Uno is a microcontroller board based on the ATmega328 ([datasheet](#)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

| [Revision 2](#) of the Uno board has a resistor pulling the BU2 HWB line to ground, making it easier to put into [DFW mode](#).

| [Revision 3](#) of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the [index of Arduino boards](#).

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: [arduino-uno-Rev3-reference-design.zip](#) (NOTE: works with Eagle 6.0 and newer)

Schematic: [arduino-uno-Rev3-schematic.pdf](#)

Notes: The Arduino reference design can use an Atmega8, 168, or 328. Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the [analogWrite\(\)](#) function.

RIWAYAT HIDUP PENULIS



Nama : Ronaldo Gabe Manik
TTL : Surabaya, 16 April 1997
Jenis Kelamin : Laki-laki
Agama : Kristen
Alamat : Jl. Tenggilis Kauman
IV/18b
Telp/HP : 081233993304
E-mail : ronaldogm@ymail.com

RIWAYAT PENDIDIKAN:

1. 2003-2009 : SD Santo Yosef Surabaya
2. 2009-2012 : SMP Santo Carolus
3. 2012-2015 : SMA Kristen Petra 5
4. 2015-sekarang : Program Studi Elektro Industri,
Departemen Teknik Elektro Otomasi,
Fakultas Vokasi, Institut Teknologi
Sepuluh Nopember, Surabaya

PENGALAMAN KERJA

1. Kerja Praktek di PT. Petrokimia Gresik
2. Kerja Praktek di PT. Industri Kereta Api (INKA) Madiun

PENGALAMAN ORGANISASI

1. Staff Departemen PSDM 2016-2017
2. Staff Hubungan Luar BEM FTI 2016-2017
3. Ketua Departemen PSDM 2017-2018